▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 39

PASO A PASO



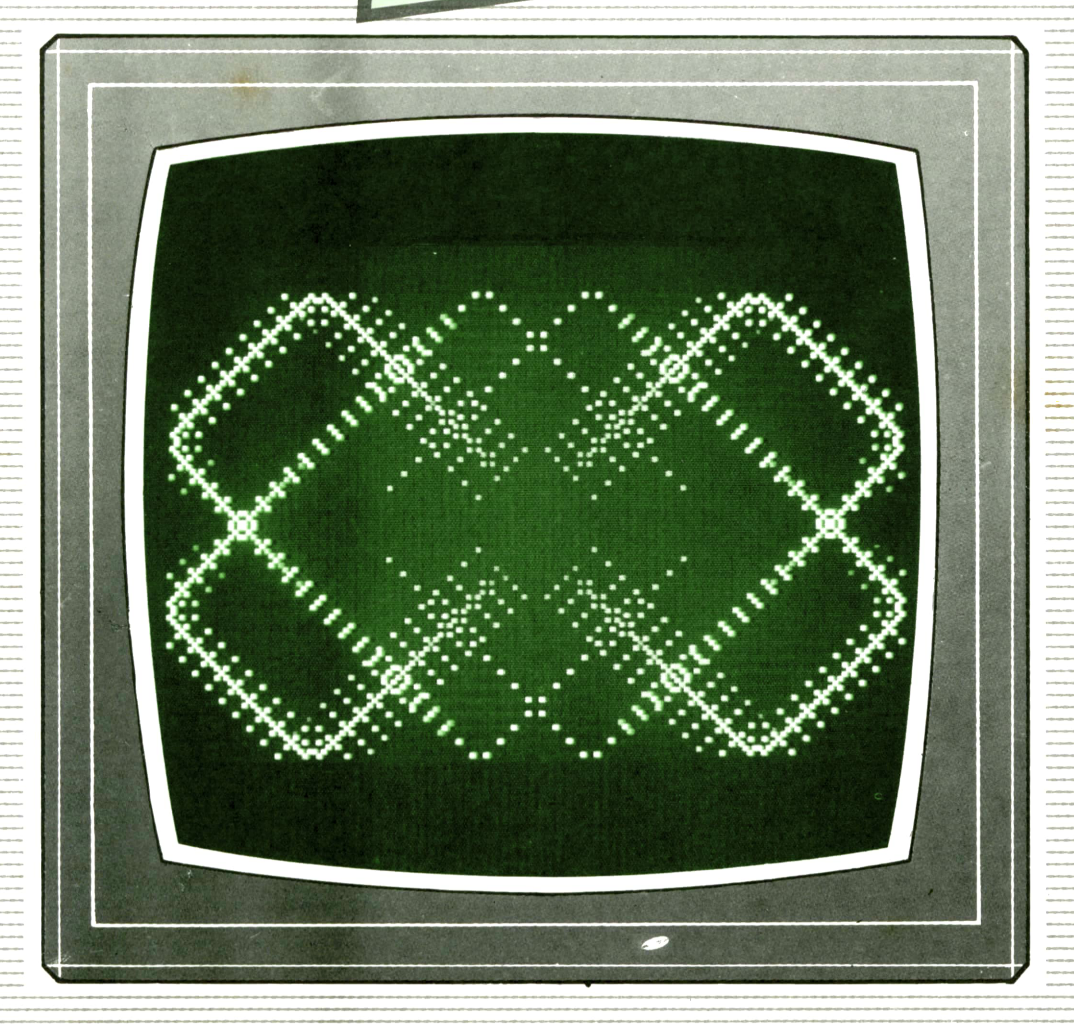
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 39

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación
y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Jorge Thomas, Técnico en Informática. OTROS LENGUAJES (ADA): Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación.

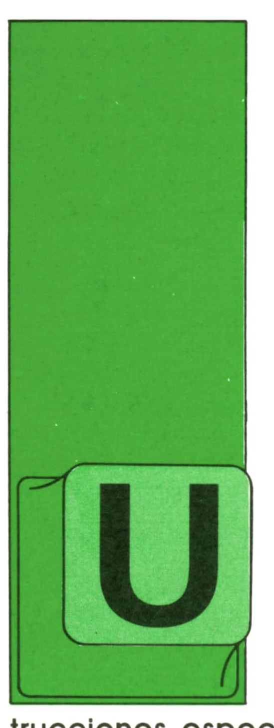
Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 455 09 99. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-191-6
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 259 73 31. 28020 Madrid.
Junio, 1988
P.V.P. Canarias: 335,–.

INDICE

BASIC



Gestión de ficheros

UN fichero es una parte de un disco o de una cinta donde se graban los datos que un programa tiene almacenados en algunas de sus variables, mediante unas instrucciones específicas, y al que puede acceder otro programa o el mismo, mediante otras instrucciones, para leer los datos grabados.

Los ficheros pueden ser de dos tipos: secuenciales o de acceso directo. Nosotros nos vamos a centrar en los ficheros secuenciales por ser los que se utilizan normalmente.

Ficheros secuenciales

Un fichero secuencial es aquel en el que los datos deben leerse en el mismo orden en que se grabaron y comenzando siempre por el primero.

Los ficheros se estructuran en registros. Un registro está constituido por todos los datos que se graban de una sola vez (en una misma instrucción). Cada instrucción que ejecute un programa para grabar datos en un fichero secuencial, creará un nuevo registro en éste, justo a continuación del último que grabó.

REGISTRO	REGISTRO	REGISTRO
GARCIA 23	NAVARRO	15 12 19

Fig. 1.

En la figura 1 podemos ver una posible estructura de un fichero secuencial con tres registros con distintos datos en cada uno.



Apertura de un fichero

En todos los programas en los que se vaya a utilizar un fichero, antes de realizar cualquier operación sobre él se debe ejecutar la instrucción OPEN (abrir), que indica al ordenador el fichero que vamos a usar y sus características.

Las características que se suelen especificar al abrir un fichero son las siguientes:

— Tipo de fichero: secuencial o de acceso directo.

— Modo para el que se abre el fichero: lectura de datos o grabación de datos.

— Número del fichero: todos los ficheros utilizados en un mismo programa deben llevar un número de identificación.

— Nombre del fichero: es el nombre con el que el fichero se graba en el disco o la cinta. Si hay varias unidades de disco o cinta hay que especificar, además del nombre, la unidad donde se encuentra el disco o la cinta donde se almacena el fichero deseado.

Veamos a continuación los formatos de la instrucción OPEN en los distintos ordenadores:

AMSTRAD

El AMSTRAD no permite utilizar en un programa más de un fichero secuencial simultáneamente. A dicho fichero le asocia siempre el número 9. Las instrucciones de apertura son:

OPENIN "nombre"

para leer los datos grabados en él, y:

OPENOUT "nombre"

para grabar datos en él.

COMMODORE

El formato es:

OPEN n.° de fichero, 1, modo, "nombre"

El 1 que aparece a continuación del n.° de fichero indica que es de tipo secuencial.
En modo pondremos O (inicial de Output) si abrimos el fichero para grabar datos e I (inicial de Input) si el fichero se abre para leer datos.
Por ejemplo, la instrucción:

OPEN, 1,1,O, "AGENDA"

indica que se abre el fichero número 1, denominado "AGENDA", para grabar datos en él.

IBM

El formato es:

OPEN "modo"#n.° de fichero, "nombre"

El significado de cada parámetro es el mismo que en el COMMODORE, pero además disponemos del modo "A", que permite abrir un fichero ya existente para añadirle nuevos datos.
El ejemplo anterior para IBM sería:

OPEN "O", #1, "AGENDA"

MSX

El formato es el siguiente:

OPEN "nombre" FOR modo AS n.° de fichero

El ejemplo anterior sería:

OPEN "AGENDA" FOR O AS TO 1

SPECTRUM

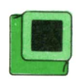
El SPECTRUM sólo permite la gestión de ficheros en microdrive (pequeñas cintas "sin fin" que trabajan de forma similar a los discos). El formato de apertura es:

OPEN #n.° de fichero; "M"; n.° de microdrive; "nombre"

El número de fichero debe ser mayor o igual que 4. La "M" indica la utilización del microdrive. El número de microdrive estará comprendido entre 1 y 8, ya que podemos utilizar hasta ocho microdrives simultáneamente.

El ejemplo anterior sería ahora:

OPEN #4;"M";1;"AGENDA"



Cierre de un fichero

Cuando hayamos terminado de realizar todas las operaciones que se hagan en un programa con un fichero, debemos cerrarlo con la instrucción CLOSE. De no hacer esta operación podemos encontrarnos con problemas de que los datos no queden grabados correctamente.
Veamos los formatos de CLOSE en los distintos ordenadores:

AMSTRAD

El formato es:

CLOSEIN

si cerramos un fichero abierto con OPENIN, y

CLOSEOUT

si cerramos un fichero abierto con OPENOUT.

COMMODORE, IBM, MSX y SPECTRUM

El formato es:

CLOSE #n.° de fichero

Por tanto, en nuestro ejemplo tendremos que escribir:

CLOSE #1

excepto en el SPECTRUM, que será:

CLOSE #4



Grabación de datos en un fichero

Para grabar datos en un fichero secuencial se utiliza la instrucción:

PRINT #n.° de fichero, <lista de variables>

La lista de variables puede contener variables numéricas y/o alfanuméricas separadas unas de otras por punto y coma (;).
Cuando un programa ejecuta una instrucción PRINT#, crea un nuevo registro en el fichero especificado, formado por todos los datos contenidos en la lista de variables.
Por ejemplo:

PRINT#1,A$;B$;C

Graba un registro en el fichero número 1, compuesto por dos datos alfanuméricos y uno numérico.



Lectura de datos de un fichero

La lectura de datos de un fichero se puede efectuar de varias formas, sin embargo, la instrucción más típica es INPUT#. Esta instrucción lee un registro completo de un fichero cada vez que se ejecute en un programa. Sin embargo, debemos tener en cuenta que para que INPUT# lea los datos de un registro éstos deben estar separados por comas; por tanto, cuando grabemos varios datos para leerlos posteriormente con INPUT# tendremos que grabar entre medias de ellos una coma. Por ejemplo:

PRINT# 1,A$;",";B$;",";C

El formato para INPUT# es el siguiente:

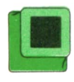
INPUT #n.° de fichero,<lista de variables>

En este caso las variables deben ir separadas por comas. Además, deben ser del mismo número y del mismo tipo que las utilizadas para grabar los datos.

Ya podemos ver un ejemplo de manejo de ficheros. El programa 1 controla el stock de diez artículos diferentes de un almacén.

```
10 REM ***********************************
20 REM * CONTROL DE STOCK DE UN ALMACEN *
30 REM ***********************************
40 DIM A(10)
50 CLS
60 PRINT TAB(17);"OPCIONES"
70 PRINT TAB(17);"________"
80 PRINT :PRINT
90 PRINT "1.  INICIALIZAR STOCK DE CADA PRODUCTO"
100 PRINT "2.  CAMBIAR STOCK DE UN PRODUCTO"
110 PRINT "3.  VISUALIZAR STOCK DE TODOS LOS PRODUCTOS"
120 PRINT "4.  GRABAR STOCK DE CADA PRODUCTO EN UN FICHERO"
130 PRINT "5.  LEER DEL FICHERO EL STOCK DE CADA PRODUCTO"
140 PRINT "6.  TERMINAR"
150 PRINT :PRINT :PRINT
160 PRINT "PULSA LA OPCION DESEADA"
170 LET R$=INKEY$:IF R$="" THEN GOTO 170
180 IF ASC(R$)<49 OR ASC(R$)>54 THEN GOTO 170
190 CLS
200 LET R=VAL(R$)
210 ON R GOSUB 1000,2000,3000,4000,5000
220 IF R<>6 THEN GOTO 50
230 END
1000 FOR I=1 TO 10
1010 PRINT "STOCK DEL PRODUCTO ";I;
1020 INPUT A(I)
1030 NEXT I
1040 RETURN
2000 INPUT "NUMERO DEL PRODUCTO DEL QUE HA HABIDO CAMBIO";P
2010 INPUT "CAMBIO DE STOCK CON SU SIGNO";C
2020 A(P)=A(P)+C
2030 RETURN
3000 FOR I=1 TO 10
3010 PRINT "STOCK DEL PRODUCTO ";I;"=";A(I)
3020 NEXT I
3030 PRINT :PRINT :PRINT
3040 PRINT "PULSA UNA TECLA PARA CONTINUAR"
3050 LET T$=INKEY$:IF T$="" THEN GOTO 3050
3060 RETURN
4000 OPEN "O",#1,"STOCK"
4010 FOR I=1 TO 10
4020 PRINT  #1,A(I)
4030 NEXT I
4040 CLOSE #1
4050 RETURN
5000 OPEN "I",#2,"STOCK"
5010 FOR I=1 TO 10
5020 INPUT #2,A(I)
5030 NEXT I
5040 CLOSE #2
5050 RETURN
```

El programa ha sido realizado en un IBM; sin embargo, resulta muy fácil adaptarlo a cualquier otro ordenador con lo explicado hasta ahora.



Programas en dispositivos de almacenamiento externo

Cuando hacemos un programa importante es conveniente grabarlo en un disco o una cinta, pues, de lo contrario, cuando apaguemos el ordenador se borrará de la memoria y si queremos utilizarlo en otra ocasión tendremos que teclearlo de nuevo.

La instrucción para grabar programas, tanto en disco como en cinta, es SAVE y tiene el siguiente formato general:

SAVE "nombre del programa"

Normalmente el nombre del programa no puede tener más de ocho o nueve caracteres y por regla general no se admiten signos ni espacios en blanco. Si tenemos varias unidades de disco o cinta tendremos que especificar delante del nombre la unidad en la que queremos grabar. Por ejemplo:

SAVE "A:NUMEROS"

grabará en la unidad A el programa llamado NUMEROS.

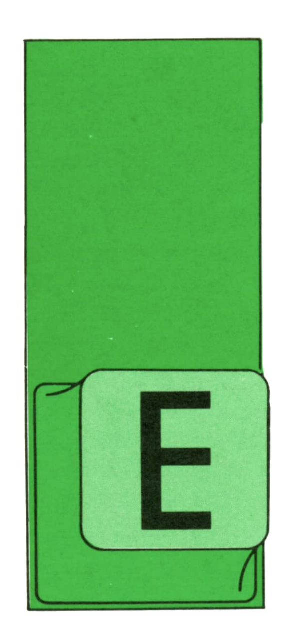
Por otra parte, para cargar en la memoria del ordenador un programa que tengamos grabado en un disco o una cinta disponemos de la instrucción LOAD con el siguiente formato:

LOAD "nombre del programa"

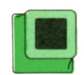
Al igual que sucedía con SAVE, si tenemos varias unidades de disco o cinta tendremos que especificar la unidad deseada.

MAQUINA Z-80

SPECTRUM, AMSTRAD, MSX



EN este capítulo final veremos algunos ejemplos más complejos como resumen de todo lo tratado anteriormente.



Programa 1: Copiar zonas de memoria

Este programa puede copiar un área de memoria en otra zona. Esto suele usarse para realizar rutinas de animación; éstas copian diferentes imágenes en la pantalla, una tras otra, en rápida sucesión para dar impresión de movimiento.

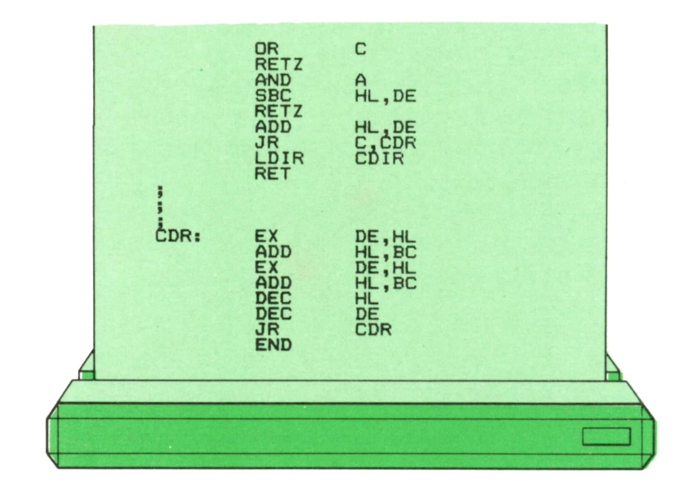
La dirección donde están los datos viene dada por COMI y donde los deja en DEST. El número de Bytes a mover está en LONG.

Lo primero que realiza el programa es la inicialización de los registros a usar con los valores adecuados: COMI en HL, DEST en DE o LONG en BC.

Tras esto comienza el bucle principal. Si el registro BC es cero, o HL se hace igual a DE, la ejecución termina. Si HL es mayor que DE, la sección se copia mediante la instrucción LDIR.

Si DE es mayor que HL, se añade BC-1 a ambos pares de registros y se copia mediante la subrutina CDR.

```
;   PROGRAMA PARA COPIAR ZONAS DE
MEMORIA
;
        .Z8Ø
COMI    EQU     23296
DEST    EQU     23298
LONG    EQU     233ØØ
;
;
;
INI     EQU     1ØØØH
;
;
INI:    LD      HL,COMI
        LD      DE,DEST
        LD      BC,LONG
CDIR:   LD      A,B
```



Programa 2: Buscar en memoria un patrón

La utilidad de este programa consiste en buscar en otro programa, o en la ROM, algunas instrucciones cuya dirección deseamos determinar.

Las variables usadas son PRINC, que marca el principio de la zona en donde buscar el patrón; LONG nos indica la longitud del patrón y PATDIR donde está; por último, BINDIR nos dará la dirección del resultado de la búsqueda.

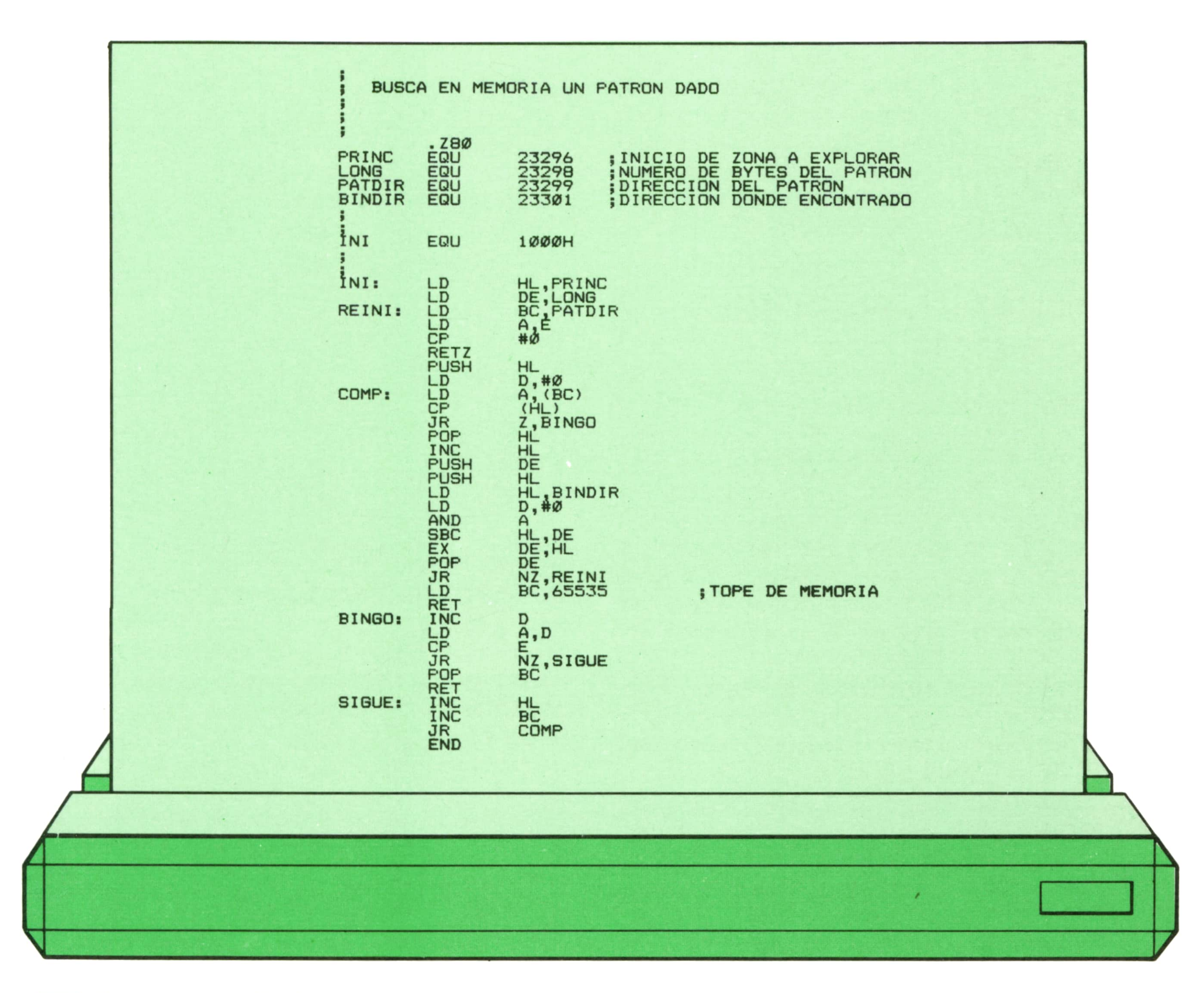
Al igual que el programa anterior, comienza con la inicialización de los valores a usar en los registros adecuados: HL con PRINT, E con LONG y PATDIR en BC. Tras esto comprueba si la longitud del patrón no es cero. Salva el registro HL en la pila del sistema y comienza a comparar el valor de la dirección indicada por HL con la indicada por BC. Si éste es el mismo, procede a saltar a BINGO, pues los patrones parecen coincidir. Si son diferentes, se extrae HL de la pila y se incrementa en uno para repetir el proceso.

Los registros DE y HL se salvan en la pila; HL se carga con BINDIR y DE con LONG,

restándose DE de HL para ver cuál sería la dirección más alta en que podría empezar el patrón, incrementándose para dar la máxima dirección donde no está el patrón dado.

Esta dirección se carga en HL y se saca de la pila el antiguo valor de DE. Si el resultado es cero, se carga BC con la máxima dirección posible (65535) para indicar que no ha habido éxito y regresa; de no ser cero, va a REINI para repetir el bucle principal.

El BINGO se comprueba si todos los caracteres del patrón coinciden con los de la memoria y de ser así se extrae BC de la pila y regresa; en caso de no coincidir totalmente, se incrementan HL y BC, regresando el bucle principal por COMP.



```
;
;   BUSCA EN MEMORIA UN PATRON DADO
;
;
;
        .Z8Ø
PRINC   EQU     23296   ;INICIO DE ZONA A EXPLORAR
LONG    EQU     23298   ;NUMERO DE BYTES DEL PATRON
PATDIR  EQU     23299   ;DIRECCION DEL PATRON
BINDIR  EQU     233Ø1   ;DIRECCION DONDE ENCONTRADO
;
INI     EQU     1ØØØH
;
INI:    LD      HL,PRINC
        LD      DE,LONG
REINI:  LD      BC,PATDIR
        LD      A,E
        CP      #Ø
        RETZ
        PUSH    HL
        LD      D,#Ø
COMP:   LD      A,(BC)
        CP      (HL)
        JR      Z,BINGO
        POP     HL
        INC     HL
        PUSH    DE
        PUSH    HL
        LD      HL,BINDIR
        LD      D,#Ø
        AND     A
        SBC     HL,DE
        EX      DE,HL
        POP     DE
        JR      NZ,REINI
        LD      BC,65535        ;TOPE DE MEMORIA
        RET
BINGO:  INC     D
        LD      A,D
        CP      E
        JR      NZ,SIGUE
        POP     BC
        RET
SIGUE:  INC     HL
        INC     BC
        JR      COMP
        END
```

Programa 3: Buscar y reemplazar

Este programa busca todas las apariciones de un patrón en la memoria y lo reemplaza por otro de igual longitud dentro de un programa.

Como en los anteriores, carga los registros con los valores a usar: HL, con la dirección del programa, y el acumulador con la longitud del patrón; se realiza a otra copia del acumulador en E para uso posterior. Si la longitud es cero, finaliza la rutina. Se prepara HL para que apunte al comienzo de la primera línea y entra en el bucle principal por RESET.

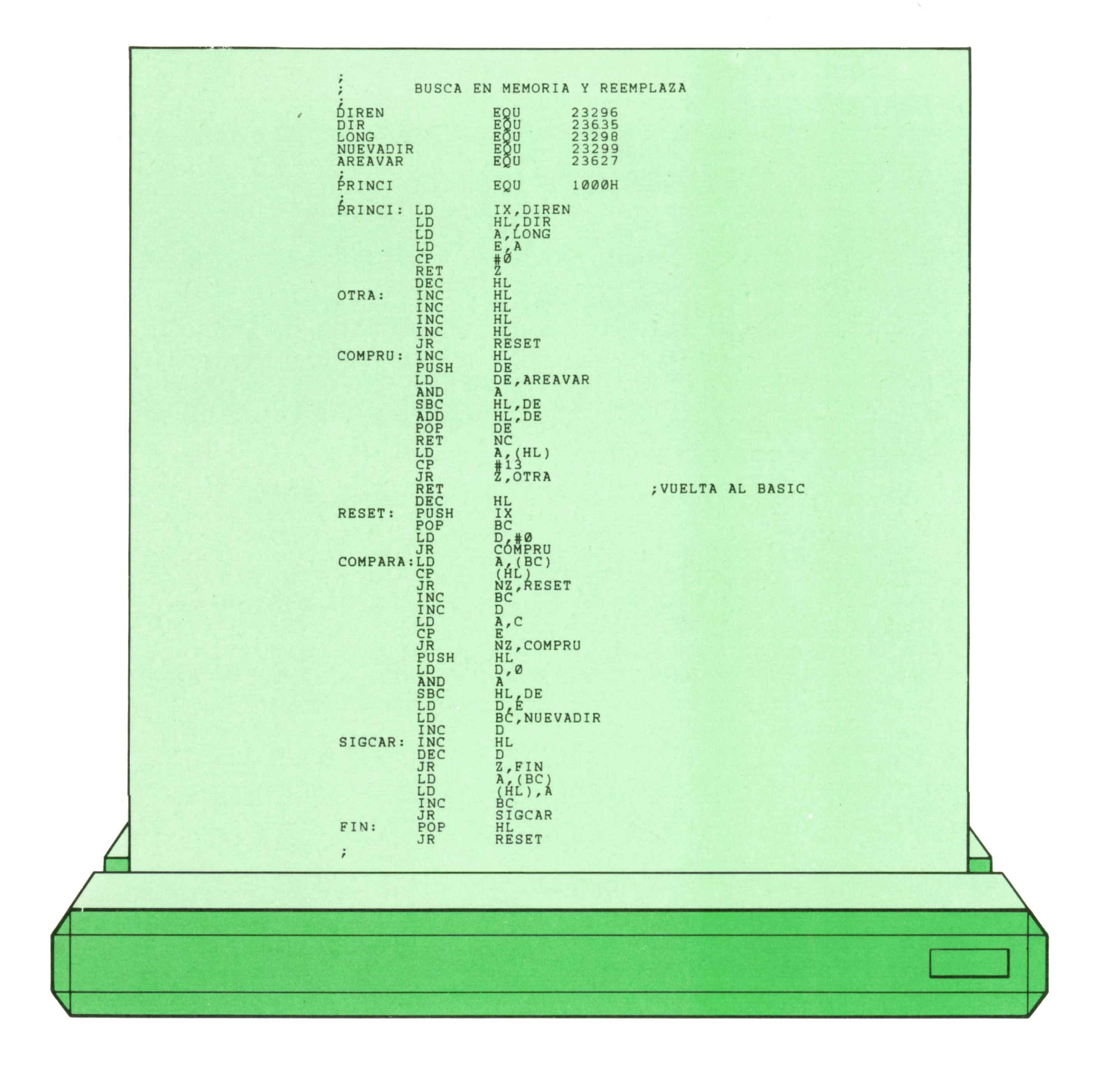
COMPRU incrementa el contenido de HL para apuntar el dato siguiente y lo compara con DE para ver si ha acabado.

Se carga el acumulador A con el siguiente dato y lo compara con el carácter de "Retorno de Carro" o "ENTER" para saltarse los números de línea, cinco caracteres.

En RESET copia en BC la dirección de la cadena buscada y se inicializa D con cero para ver el número de caracteres de la cadena hallada. Tras esto, regresa a COMPRU.

COMPARA ve si el patrón hallado coincide en toda su longitud con el dato; de no ser así, vuelve al bucle COMPRU.

Si coinciden totalmente los patrones realiza la sustitución por el nuevo valor en el bucle SIG CAR. Al acabar en FIN comienza la búsqueda de las siguientes apariciones en RESET.



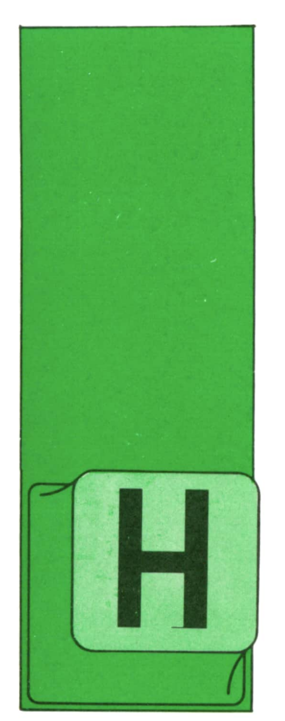
```
;
;         BUSCA EN MEMORIA Y REEMPLAZA
;
DIREN             EQU      23296
DIR               EQU      23635
LONG              EQU      23298
NUEVADIR          EQU      23299
AREAVAR           EQU      23627
;
PRINCI            EQU      1000H
;
PRINCI:  LD       IX,DIREN
         LD       HL,DIR
         LD       A,LONG
         LD       E,A
         CP       #0
         RET      Z
         DEC      HL
OTRA:    INC      HL
         INC      HL
         INC      HL
         INC      HL
         JR       RESET
COMPRU:  INC      HL
         PUSH     DE
         LD       DE,AREAVAR
         AND      A
         SBC      HL,DE
         ADD      HL,DE
         POP      DE
         RET      NC
         LD       A,(HL)
         CP       #13
         JR       Z,OTRA
         RET                            ;VUELTA AL BASIC
         DEC      HL
RESET:   PUSH     IX
         POP      BC
         LD       D,#0
         JR       COMPRU
COMPARA:LD        A,(BC)
         CP       (HL)
         JR       NZ,RESET
         INC      BC
         INC      D
         LD       A,C
         CP       E
         JR       NZ,COMPRU
         PUSH     HL
         LD       D,0
         AND      A
         SBC      HL,DE
         LD       D,E
         LD       BC,NUEVADIR
         INC      D
SIGCAR:  INC      HL
         DEC      D
         JR       Z,FIN
         LD       A,(BC)
         LD       (HL),A
         INC      BC
         JR       SIGCAR
FIN:     POP      HL
         JR       RESET
;
```

PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS



Definidor de gráficos para IBM



HEMOS visto en algunos tomos de esta colección programas para definir caracteres y SPRITES en los distintos ordenadores que contemplamos. Como en el IBM no se pueden definir caracteres, pero sí podemos definir y almacenar una pequeña zona de la pantalla y moverla como si de un SPRITE se tratase, proponemos el siguiente programa para todos aquellos que necesiten definir figuras y gráficos para sus programas.

Con este programa podemos definir cualquier gráfico que tenga como máximo 19 × 40 puntos. Una vez definido podemos almacenarlo o imprimirlo. Para ello contamos con una serie de comandos disponibles desde las teclas de función. Estos son:

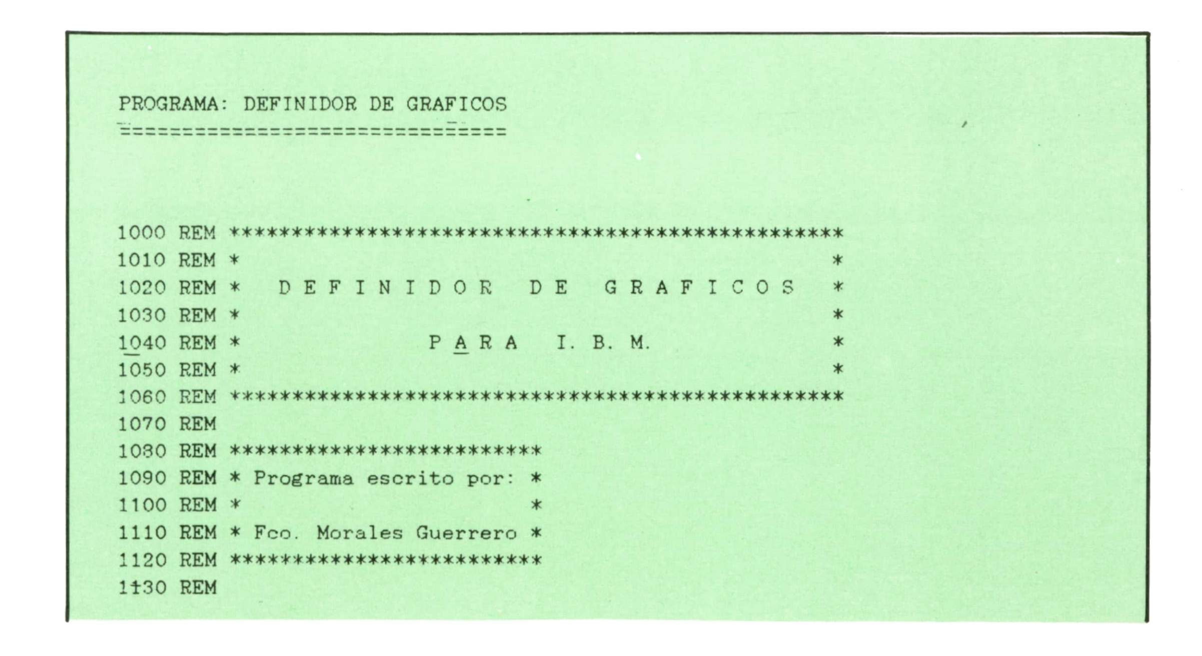
F1. Nos permite grabar el gráfico en formato interno. Si grabamos el gráfico con esta opción, cuando necesitemos modificarlo, o terminarlo, pulsando la opción F3 podemos leerlo y visualizarlo en pantalla. Este formato de grabación no es muy útil al programador.

F2. Nos graba el gráfico que hemos definido como una sucesión de números que se encuentran en unas líneas DATA. Podemos elegir qué número han de tener dichas líneas DATA. Esta opción se utiliza para que el usuario no tenga que teclear dichas DATA y para que realizando un simple MERGE entre su programa y el fichero creado, disponga del gráfico definido.

F3. Con esta opción podemos leer un fichero que contiene la definición de un gráfico y que nosotros hemos grabado anteriormente con la opción F1.

F4. Nos permite imprimir en impresora las líneas DATA con la sucesión de números que nos definen un gráfico. Podemos elegir el número de líneas por el que queremos que empiecen dichas líneas.

F9. Terminamos la sesión de edición y salimos del programa.



```
PROGRAMA: DEFINIDOR DE GRAFICOS
===============================


1000 REM ***************************************************
1010 REM *                                                 *
1020 REM *   D E F I N I D O R   D E   G R A F I C O S    *
1030 REM *                                                 *
1040 REM *              P A R A   I. B. M.                 *
1050 REM *                                                 *
1060 REM ***************************************************
1070 REM
1080 REM *************************
1090 REM * Programa escrito por: *
1100 REM *                       *
1110 REM * Fco. Morales Guerrero *
1120 REM *************************
1130 REM
```

```
1140 REM ********************************
1150 REM *                              *
1160 REM * (c) Ediciones Siglo Cultural *
1170 REM *                              *
1180 REM * (c) 1987.                    *
1190 REM *                              *
1200 REM ********************************
1210 REM
1220 REM *********************
1230 REM *** INSTRUCCIONES ***
1240 REM *********************
1250 REM
1260 KEY OFF
1270 SCREEN 2
1280 WIDTH 80
1290 CLS
1300 PRINT TAB(30);"DEFINIDOR DE GRAFICOS"
1310 PRINT TAB(29);"======================="
1320 PRINT
1330 PRINT
1340 PRINT " Usa las teclas del cursor (8,2,4,6) para moverte por la parrilla."
1350 PRINT
1360 PRINT " Pulsa la barra espaciadora para encender o apagar un punto."
1370 PRINT
1380 PRINT " Pulsa F1 para grabar lo definido en formato interno."
1390 PRINT
1400 PRINT " Pulsa F2 para grabar lo definido como lineas DATAS."
1410 PRINT
1420 PRINT " Pulsa F3 para leer un grafico ya definido desde disco."
1430 PRINT
1440 PRINT " Pulsa F4 para imprimir el grafico como lineas DATA."
1450 PRINT
1460 PRINT " Pulsa F9 para salir del programa."
1470 PRINT
1480 PRINT " Solo puede definirse un grafico cada vez."
1490 PRINT
1500 PRINT
1510 PRINT " PULSA UNA TECLA PARA COMENZAR"
1520 LET A$=INKEY$
1530 IF A$="" THEN GOTO 1520
1540 REM
1550 REM ************************
1560 REM *** ENTRADA DE DATOS ***
1570 REM ************************
1580 REM
1590 CLS
1600 PRINT TAB(30);"DEFINIDOR DE GRAFICOS"
1610 PRINT TAB(29);"======================="
1620 PRINT
1630 PRINT
1640 PRINT " Cual va a ser la altura del grafico? (Maximo 19)."
1650 PRINT
1660 INPUT " ===> ",H
1670 IF H<1 OR H>19 THEN PRINT:PRINT "VALOR NO VALIDO":GOTO 1650
1680 PRINT
1690 PRINT
1700 PRINT " Cual va a se la anchura del grafico? (Maximo 40)."
1710 PRINT
1720 INPUT "===> ",L
1730 IF L<1 OR L>40 THEN PRINT:PRINT "VALOR NO VALIDO":GOTO 1710
1740 REM
1750 REM **********************
1760 REM *** INICIALIZACION ***
1770 REM **********************
1780 REM
```

```
1790 KEY 1,""
1800 FOR I=1 TO 9
1810    KEY I,""
1820 NEXT I
1830 ON KEY(1) GOSUB 2570
1840 ON KEY(2) GOSUB 2840
1850 ON KEY(3) GOSUB 3220
1860 ON KEY(4) GOSUB 3770
1870 ON KEY(9) GOSUB 4150
1880 FOR I=1 TO 9
1890    KEY(I) ON
1900 NEXT I
1910 DIM A%(2*L)
1920 GET(0,0)-(L,H),A%
1930 REM
1940 REM ***********************************************
1950 REM *** DIBUJO DE LA PARRILLA Y DE LA PANTALLA ***
1960 REM ***********************************************
1970 REM
1980 CLS
1990 FOR I=1 TO H
2000    FOR J=1 TO L
2010       PRINT CHR$(254);
2020    NEXT J
2030    PRINT
2040 NEXT I
2050 COLOR ,,1
2060 LINE(0,183)-(639,191),,BF
2070 LOCATE 24,1
2080 PRINT "F1---SAVE FORMATO INTERNO   F2---SAVE COMO DATA  F3---LOAD  F4---PRI
NT  F9---FIN";
2090 COLOR ,,0
2100 LINE (400,0)-(410+L,10+H),,B
2110 LET X1=405
2120 LET Y1=5
2130 LET X2=X1+L
2140 LET Y2=Y1+H
2150 LET X=1
2160 LET Y=1
2170 REM
2180 REM *************************
2190 REM *** PROGRAMA PRINCIPAL ***
2200 REM *************************
2210 REM
2220 LET A$=INKEY$
2230 IF A$=" " THEN GOTO 2440
2240 LET DX=0
2250 LET DY=0
2260 IF LEN(A$)=2 THEN LET A$=MID$(A$,2,1)
2270 IF A$=CHR$(77) THEN LET DX=1:LET DY=0
2280 IF A$=CHR$(75) THEN LET DX=-1:LET DY=0
2290 IF A$=CHR$(72) THEN LET DY=-1:LET DX=0
2300 IF A$=CHR$(80) THEN LET DY=1:LET DX=0
2310 LET C=POINT(X1+X,Y1+Y)
2320 IF C=1 THEN LOCATE Y,X:PRINT CHR$(219)
2330 IF C=0 THEN LOCATE Y,X:PRINT CHR$(254)
2340 Y=Y+DY
2350 X=X+DX
2360 IF X>L THEN LET X=L
2370 IF X<1 THEN LET X=1
2380 IF Y>H THEN LET Y=H
2390 IF Y<1 THEN LET Y=1
2400 LOCATE Y,X
2410 PRINT " "
2420 GOTO 2220
```

```
2430 GET (0,0)-(L,H),A%
2440 REM
2450 REM **********************************
2460 REM * FIJACION Y BORRADO DE PUNTOS ***
2470 REM **********************************
2480 REM
2490 LET C=-POINT(X1+X,Y1+Y)
2500 LET C=-NOT C
2510 PSET(X1+X,Y1+Y),C
2520 GET(X1,Y1)-(X2,Y2),A%
2530 IF C=0 THEN LOCATE Y,X:PRINT CHR$(254)
2540 IF C=1 THEN LOCATE Y,X:PRINT CHR$(219)
2550 GOTO 2220
2560 GOTO 2560
2570 REM
2580 REM *******************************
2590 REM *** SAVE EN FORMATO INTERNO ***
2600 REM *******************************
2610 REM
2620 KEY(1) STOP
2630 LINE(0,159)-(639,167),,BF
2640 LOCATE 21,29
2650 COLOR ,,1
2660 PRINT "SAVE EN FORMATO INTERNO"
2670 COLOR ,,0
2680 GOSUB 4470 : REM PETICION DE NOMBRE
2690 LET N$=N$+".GRP"
2700 LOCATE 22,1
2710 PRINT SPACE$(80)
2720 LOCATE 22,1
2730 PRINT "GRABANDO CON FORMATO INTERNO: ";N$
2740 OPEN N$ FOR OUTPUT AS #1
2750 PRINT #1,L
2760 PRINT #1,H
2770 FOR I=0 TO 2*L
2780    PRINT#1,A%(I)
2790 NEXT I
2800 CLOSE #1
2810 GOSUB 4560 : REM FIN DE FUNCION
2820 KEY(1) ON
2830 RETURN
2840 REM
2850 REM *****************************
2860 REM *** SAVE COMO LINEAS DATA ***
2870 REM *****************************
2880 REM
2890 KEY(2) STOP
2900 LINE(0,159)-(639,167),,BF
2910 LOCATE 21,28
2920 COLOR ,,1
2930 PRINT "SAVE COMO LINEAS DATA"
2940 COLOR ,,0
2950 GOSUB 4470 : REM PETICION DE NOMBRE
2960 LET N$=N$+".BAS"
2970 OPEN N$ FOR OUTPUT AS #1
2980 LOCATE 22,1
2990 PRINT SPACE$(80)
3000 LOCATE 22,1
3010 INPUT "DESDE QUE LINEA ==> ";LI
3020 LOCATE 22,1
3030 PRINT SPACE$(80)
3040 LOCATE 22,1
3050 PRINT "GRABANDO COMO LINEAS DATA: ";N$
3060 PRINT #1,LI;"DATA";L;",";H
3070 LET LI=LI+10
```

```
3080 FOR I=0 TO 2*L STEP 10
3090    PRINT #1,LI;"DATA";
3100    N=8
3110    IF I+9>2*L THEN LET N=2*L-I-1
3120    FOR J=I TO I+N
3130       PRINT #1,A%(J);",";
3140    NEXT J
3150    PRINT #1,A%(J)
3160    LET LI=LI+10
3170 NEXT I
3180 CLOSE #1
3190 GOSUB 4560 : REM OPERACION TERMINADA
3200 KEY(2) ON
3210 RETURN
3220 REM
3230 REM ********************************
3240 REM *** LOAD EN FORMATO INTERNO ***
3250 REM ********************************
3260 REM
3270 KEY(3) STOP
3280 LINE(0,159)-(639,167),,BF
3290 LOCATE 21,29
3300 COLOR ,,1
3310 PRINT "LOAD EN FORMATO INTERNO"
3320 COLOR ,,0
3330 GOSUB 4470 : REM PETICION DE NOMBRE
3340 LET N$=N$+".GRP"
3350 LOCATE 22,1
3360 PRINT SPACE$(80)
3370 LOCATE 22,1
3380 PRINT "CARGANDO CON FORMATO INTERNO: "N$
3390 OPEN N$ FOR INPUT AS #1
3400 ERASE A%
3410 INPUT #1,L
3420 INPUT #1,H
3430 DIM A%(2*L)
3440 FOR I=0 TO 2*L
3450    INPUT #1,A%(I)
3460 NEXT I
3470 CLOSE #1
3480 LOCATE 22,1
3490 PRINT SPACE$(80)
3500 LOCATE 22,1
3510 PRINT "ESPERE UN MOMENTO"
3520 FOR I=1 TO 19
3530    LOCATE I,1
3540    PRINT SPACE$(80)
3550 NEXT I
3560 LOCATE 1,1
3570 FOR I=1 TO H
3580    FOR J=1 TO L
3590       PRINT CHR$(254);
3600    NEXT J
3610    PRINT
3620 NEXT I
3630 LINE(400,0)-(410+L,10+H),,B
3640 LET X2=X1+L
3650 LET Y2=Y1+H
3660 PUT(X1,Y1),A%
3670 FOR I=1 TO H
3680    FOR J=1 TO L
3690       IF POINT(X1+J,Y1+I)=1 THEN LOCATE I,J:PRINT CHR$(219)
3700    NEXT J
3710 NEXT I
3720 LET X=1
```

```
3730 LET Y=1
3740 GOSUB 4560 : REM FIN DE FUNCION
3750 KEY(3) ON
3760 RETURN
3770 REM
3780 REM *************************************
3790 REM *** IMPRESION COMO LINEAS DE DATA ***
3800 REM *************************************
3810 REM
3820 KEY(4) STOP
3830 LINE(0,159)-(639,167),,BF
3840 LOCATE 21,26
3850 COLOR ,,1
3860 PRINT "IMPRESION COMO LINEAS DE DATA"
3870 COLOR ,,0
3880 LOCATE 22,1
3890 PRINT "COLOQUE LA IMPRESORA Y PULSE ENTER"
3900 LET A$=INKEY$
3910 IF A$<>CHR$(13) THEN GOTO 3900
3920 LOCATE 22,1
3930 PRINT SPACE$(80)
3940 LOCATE 22,1
3950 INPUT "A PARTIR DE QUE LINEA EMPIEZO ===> ",LI
3960 LOCATE 22,1
3970 PRINT SPACE$(80)
3980 LOCATE 22,1
3990 PRINT "IMPRIMIENDO ... "
4000 LPRINT LI;"DATA";L;",";H
4010 LET LI=LI+10
4020 FOR I=0 TO 2*L STEP 10
4030    LPRINT LI;"DATA";
4040    LET N=8
4050    IF I+N>2*L THEN LET N=2*L-I-1
4060    FOR J=I TO I+N
4070       LPRINT A%(J);",";
4080    NEXT J
4090    LPRINT A%(J)
4100    LET LI=LI+10
4110 NEXT I
4120 GOSUB 4560 : REM FIN DE FUNCION
4130 KEY(4) ON
4140 RETURN
4150 REM
4160 REM ***************************
4170 REM *** SALIDA DEL PROGRAMA ***
4180 REM ***************************
4190 REM
4200 KEY(9) STOP
4210 LINE(0,159)-(639,167),,BF
4220 LOCATE 21,32
4230 COLOR ,,1
4240 PRINT "SALIR DEL PROGRAMA"
4250 COLOR ,,0
4260 LOCATE 22,1
4270 PRINT "ESTAS SEGURO? (S/N)"
4280 LET A$=INKEY$
4290 IF A$="" THEN GOTO 4280
4300 IF A$="S" OR A$="s" THEN GOTO 4360
4310 IF A$="N" OR A$="n" THEN GOTO 4330
4320 GOTO 4280
4330 GOSUB 4560 : REM FIN DE FUNCION
4340 KEY(9) ON
4350 RETURN
4360 REM
```

```
4370 REM ************************
4380 REM *** FIN DEL PROGRAMA ***
4390 REM ************************
4400 REM
4410 SCREEN 0
4420 PRINT "ADIOS ..."
4430 FOR I=1 TO 10
4440    PRINT
4450 NEXT I
4460 END
4470 REM
4480 REM **************************
4490 REM *** PETICION DE NOMBRE ***
4500 REM **************************
4510 REM
4520 LOCATE 22,1
4530 INPUT "NOMBRE ( SIN EXTENSION ) ==> ";N$
4540 IF LEN(N$)>8 THEN GOTO 4520
4550 RETURN
4560 REM
4570 REM **********************
4580 REM *** FIN DE FUNCION ***
4590 REM **********************
4600 REM
4610 LOCATE 21,1
4620 PRINT SPACE$(160)
4630 LINE(0,159)-(639,167),,BF
4640 LOCATE 21,24
4650 COLOR ,,1
4660 PRINT "FUNCION REALIZADA. PULSE UNA TECLA"
4670 COLOR ,,0
4680 LET A$=INKEY$
4690 IF A$="" THEN GOTO 4680
4700 LOCATE 20,1
4710 FOR I=20 TO 22
4720    PRINT SPACE$(80)
4730 NEXT I
4740 RETURN
```

Al principio del programa el ordenador nos pedirá la anchura y la altura que va a tener el gráfico. Una vez que se lo hemos dicho, nos aparece una nueva pantalla que, para uso nuestro, vamos a considerar dividida en tres zonas. Estas son:

Zona 1. Se encuentra en la esquina superior izquierda de la pantalla. Aquí nos aparecerá la cuadrícula donde diseñaremos nuestro gráfico.

Zona 2. Se encuentra en la esquina superior derecha de la pantalla. En ella nos aparece un cuadrado que está vacío. No por mucho tiempo. Según vayamos definiendo nuestro gráfico, dentro del cuadrado aparecerá dicho gráfico tal y como lo veremos cuando lo utilicemos en nuestros programas.

Zona 3. Esta zona ocupa las cuatro últimas líneas de la pantalla. En ellas, aparte de encontrarse un pequeño resumen de los comandos de que disponemos, podremos ver toda la información con respecto al programa. En esta zona será donde el usuario y el ordenador intercambien datos e informaciones.

Visto esto ya estamos listos para empezar con la definición de nuestro primer gráfico. Si nos fijamos en la cuadrícula, podemos ver que en su esquina superior izquierda se encuentra un cursor parpadeante. Dicho cursor será el encargado

de iluminar o apagar puntos en la cuadrícula. Para moverlo utilizaremos las teclas del cursor que tenemos en el teclado numérico a nuestra derecha.

8. Mover el cursor hacia arriba.
2. Mover el cursor hacia abajo.
4. Mover el cursor hacia la izquierda.
6. Mover el cursor hacia la derecha.

Para que no haya problemas a la hora de utilizar las teclas del cursor, hay que recordar que el teclado numérico se tiene que encontrar en modo cursor, y no en modo numérico. Para ello, tenemos que comprobar que la luz que hay junto a NUM LOCK está apagada.

Una vez que podemos mover el cursor por la pantalla, para iluminar el punto sobre el que se encuentra el cursor basta con pulsar la barra de espacio. Si lo que queremos es apagar el punto sobre el que se encuentra el cursor, también tenemos que pulsar la barra de espacio.

Cuando pulsamos la opción F2 para grabar un fichero que contiene las líneas DATA con la definición de nuestro gráfico, la estructura es la siguiente.

— En la primera línea nos aparecen sólo dos números. Estos son el ancho y el largo, en puntos, que tiene el gráfico que hemos definido. Estos dos números no son importantes y se graba por si son útiles al usuario.

— A continuación vienen una serie de líneas con números. Estos son los números que tenemos que almacenar en un vector entero para utilizar el gráfico que están definiendo.

— Por último, aparecen una serie de líneas todas llenas de ceros. Estas líneas no sirven para nada y el usuario puede anularlas. Se han puesto estas líneas para que se pueda apreciar perfectamente cuándo ha terminado la definición del gráfico.

El programa que aparece a continuación nos muestra un ejemplo de cómo utilizar el grupo de líneas. Para poder usar un gráfico definido.

```
PROGRAMA DEMO PARA IBM
======================
1000 REM ***************************************************
1010 REM *    PROGRAMA DEMO PARA VER COMO UTILIZAR LAS     *
1020 REM * LINEAS DATAS CON LA DEFINICION DE LOS GRAFICOS *
1030 REM ***************************************************
1040 REM
1050 CLS
1060 SCREEN 2
1070 DIM A%(40)
1080 RESTORE 9050
1090 READ L,H
1100 FOR I=0 TO 39
1110    READ A%(I)
1120 NEXT I
1130 LOCATE 14,1
1140 PRINT STRING$(80,219)
1150 FOR I=0 TO 596 STEP 4
1160    FOR J=1 TO 2
1170       PUT(I,93),A%
1180    NEXT J
1190 NEXT I
1200 END
9000 REM
9010 REM **********************************************
9020 REM * LINEAS DATA CON LA DEFINICION DEL GRAFICO *
9030 REM **********************************************
9040 REM
9050 DATA 40 , 10
9060 DATA 41 , 11 , 0 , 0 , 0 ,-241 ,-1 , 0 , 276 ,-32511
9070 DATA 128 , 296 , 24576 , 96 , 336 , 8128 , 255 , 336 , 224 , 1
9080 DATA-16047 , 256 , 193 , 8490 , 512 ,-32733 ,-8419 ,-513 , 220 ,-8189
9090 DATA 768 , 224 ,-16383 , 256 , 192 , 0 , 0 , 0 , 0 , 0
```



Monitor de memoria para el COMMODORE 64

A la hora de trabajar en código máquina, como debemos de usar la memoria del ordenador y tenemos que realizar muchas funciones con ella, se hace necesario la utilización de un programa que nos permita mover BYTES, ver qué hay en una serie de posiciones de memoria, copiar BYTES, imprimirlos, cambiar el contenido de algunas posiciones de memoria, etc. Con este programa podremos hacer todo esto de una forma muy cómoda.

```
PROGRAMA: MONITOR PARA COMMODORE
================================

1000 REM **********************
1010 REM * MONITOR DE MEMORIA *
1020 REM **********************
1030 REM
1040 REM ********************************
1050 REM *                              *
1060 REM * (c) Ediciones Siglo Cultural *
1070 REM *                              *
1080 REM * (c) 1987                     *
1090 REM *                              *
1100 REM ********************************
1110 REM
1120 REM ******************
1130 REM * INICIALIZACION *
1140 REM ******************
1150 REM
1160 OPEN 1,0,1
1170 OPEN 2,4,2
1180 LET CH=1
1190 LET BA=10
1200 LET MP=0
1210 LET MO=2
1220 LET SW=1
1230 REM
1240 REM **********************
1250 REM * PROGRAMA PRINCIPAL *
1260 REM **********************
1270 REM
1280 PRINT CHR$(147)
1290 POKE 214,1:POKE 211,22
1300 PRINT "ESTOY EN BASE ";BA
1310 GOSUB 1570
1320 GET K$
1330 IF K$="" THEN GOTO 1320
1340 IF K$="1" THEN LET MP=MP-1
1350 IF MP<0 THEN LET MP=65535!
1360 IF K$="2" THEN LET MP=MP+1
1370 IF MP>65535! THEN LET MP=0
1380 IF K$="8" THEN LET MP=MP-10
1390 IF MP<0 THEN LET MP=65535!+MP
1400 IF K$="9" THEN LET MP=MP+10
1410 IF MP>65535! THEN LET MP=MP-65535!
1420 IF K$>"0" AND K$<":" THEN GOSUB 1570:GOTO 1320
1430 IF K$="0" THEN GOSUB 4490:INPUT "VALOR A POKEAR = ";P:IF P>255 OR P<0 THEN
GOTO 1430
1440 IF K$="0" THEN POKE MP,P:GOSUB 4490:GOTO 1280
1450 IF K$="A" THEN GOSUB 3890:GOTO 1280
1460 IF K$="B" THEN GOSUB 2190:GOTO 1320
1470 IF K$="N" THEN GOSUB 2380:GOTO 1320
```

```
1480 IF K$="J" THEN GOSUB 2550:GOTO 1280
1490 IF K$="L" THEN GOSUB 2710:GOTO 1280
1500 IF K$="S" THEN GOSUB 3040:GOTO 1280
1510 IF K$="M" THEN GOSUB 3210:GOTO 1280
1520 IF K$="X" THEN GOSUB 3320:GOTO 1280
1530 IF K$="P" THEN GOSUB 1820:GOTO 1280
1540 IF K$="T" THEN GOSUB 3470:GOTO 1280
1550 IF K$="D" THEN LET SW=-SW:PRINT CHR$(147):GOTO 1280
1560 GOTO 1320
1570 REM
1580 REM *************************
1590 REM * DISPLAY DE LA MEMORIA *
1600 REM *************************
1610 REM
1620 IF SW=-1 THEN GOTO 4200
1630 POKE 214,0:POKE 211,0
1640 FOR A=MP-10 TO MP+11
1650   LET B=A
1660   IF A>65535! THEN LET A=A-65536!
1670   IF A<0 THEN LET A=65536!+A
1680   LET N$=MID$(STR$(A),2)
1690   LET P$=MID$(STR$(PEEK(A)),2)
1700   IF MO=1 THEN LET NU=VAL(P$):GOSUB 3650
1710   IF LEN(N$)<5 THEN LET N$="0"+N$:GOTO 1710
1720   IF MO=2 THEN IF LEN(P$)<3 THEN LET P$="0"+P$:GOTO 1720
1730   IF MO=3 THEN LET NU=VAL(P$):GOSUB 2010
1740   PRINT#CH," ";N$;" ";P$
1750   LET A=B
1760 NEXT A
1770 POKE 214,10:POKE 211,0:PRINT ">"
1780 IF MO=1 THEN POKE 214,10:POKE 211,15:PRINT "<"
1790 IF MO=2 THEN POKE 214,10:POKE 211,10:PRINT "<"
1800 IF MO=3 THEN POKE 214,10:POKE 211,9:PRINT "<"
1810 RETURN
1820 REM
1830 REM ************************
1840 REM * SALIDA POR IMPRESORA *
1850 REM ************************
1860 REM
1870 GOSUB 4490:INPUT "DESDE LA DIR. ";I:GOSUB 4490
1880 IF I<0 THEN GOTO 1870
1890 GOSUB 4490:INPUT "HASTA LA DIR. ";F:GOSUB 4490
1900 IF F<I OR F>65535! THEN GOTO 1890
1910 LET A=MP
1920 FOR M=I TO F STEP 22
1930    LET MP=M
1940    LET CH=2
1950    GOSUB 1570
1960 NEXT M
1970 LET MP=A
1980 LET CH=1
1990 GOSUB 1570
2000 RETURN
2010 REM
2020 REM **********************************
2030 REM * PASO DE DECIMAL A HEXADECIMAL *
2040 REM **********************************
2050 REM
2060 LET P$=""
2070 FOR C=1 TO 0 STEP -1
2080    LET N=INT(NU/16^C)
2090    LET NU=NU-N*16^C
2100    IF N<10 THEN LET P$=P$+MID$(STR$(N),2,1)
```

```
2110     IF N=10 THEN LET P$=P$+"A"
2120     IF N=11 THEN LET P$=P$+"B"
2130     IF N=12 THEN LET P$=P$+"C"
2140     IF N=13 THEN LET P$=P$+"D"
2150     IF N=14 THEN LET P$=P$+"E"
2160     IF N=15 THEN LET P$=P$+"F"
2170 NEXT C
2180 RETURN
2190 REM
2200 REM ********************
2210 REM * BUSQUEDA DE BYTES *
2220 REM ********************
2230 REM
2240 GOSUB 4490
2250 INPUT "CUANTOS BUSCO? (1-5) ";B
2260 GOSUB 4490
2270 IF B>5 OR B<1 THEN GOTO 2240
2280 FOR A=1 TO B
2290     PRINT CHR$(147)
2300     POKE 214,21:POKE 211,0
2310     PRINT "DAME EL NUMERO ";A
2320     GOSUB 4490
2330     INPUT "NUMERO > ";C(A)
2340     GOSUB 4490
2350     IF C>255 OR C<0 THEN GOTO 2330
2360 NEXT A
2370 PRINT CHR$(147)
2380 GOSUB 4490
2390 PRINT "ESPERE UN MOMENTO. BUSCANDO"
2400 FOR I=0 TO 65535!
2410     LET NN=0
2420     FOR J=0 TO B-1
2430         IF PEEK(I+J)=C(J+1) THEN NN=NN+1
2440     NEXT J
2450     IF NN=B THEN LET A=I:GOTO 2470
2460 NEXT I
2470 LET MP=A
2480 PRINT CHR$(147)
2490 GOSUB 4490
2500 PRINT "ENCONTRADO. PULSE UNA TECLA"
2510 GET A$
2520 IF A$="" THEN GOTO 2510
2530 GOSUB 1570
2540 RETURN
2550 REM
2560 REM ************************************
2570 REM * SALTO A UNA DIRECCION DE MEMORIA *
2580 REM ************************************
2590 REM
2600 GOSUB 4490
2610 INPUT "A QUE DIRECCION SALTO? (0-65535) ";J
2620 GOSUB 4490
2630 IF J<0 OR J>65535! THEN GOTO 2600
2640 GOSUB 4490
2650 PRINT "ESTAS SEGURO? (S/N)"
2660 GET A$
2670 IF A$="" THEN GOTO 2660
2680 IF A$="S" OR A$="s" THEN SYS J
2690 IF A$<>"N" AND A$<>"n" THEN GOTO 2660
2700 GOTO 4150
2710 REM
2720 REM **********************
2730 REM * CARGAR BYTES (LOAD) *
2740 REM **********************
```

```
2750 REM
2760 GOSUB 2870
2770 IF A$="D" OR A$="d" THEN OPEN 3,8,15,D$:GOTO 2800
2780 IF A$<>"C" AND A$<>"c" THEN GOTO 2760
2790 OPEN 3,1,0,D$
2800 INPUT #3,NN
2810 FOR Z=0 TO NN-1
2820    INPUT #3,A
2830    POKE D+Z,A
2840 NEXT Z
2850 CLOSE 3
2860 RETURN
2870 REM
2880 REM *******************
2890 REM * CASETE O DISCO? *
2900 REM *******************
2910 REM
2920 GOSUB 4490
2930 PRINT "CASETE O DISCO (C/D) ?"
2940 GET A$
2950 IF A$="" THEN GOTO 2940
2960 GOSUB 4490
2970 INPUT "NOMBRE? (1-8 CHRS.) ";D$
2980 IF LEN(D$)>8 OR LEN(D$)=0 THEN GOTO 2960
2990 GOSUB 4490
3000 INPUT "DESPLAZAMIENTO ? ";D
3010 IF D<0 OR D>65535! THEN GOTO 2990
3020 GOSUB 4490
3030 RETURN
3040 REM
3050 REM ***********************
3060 REM * GRABAR BYTES (SAVE) *
3070 REM ***********************
3080 REM
3090 GOSUB 2870
3100 GOSUB 4490
3110 INPUT "LONGITUD? ";L
3120 IF L>35535! OR L<0 THEN GOTO 3100
3130 IF A$="D" OR A$="d" THEN OPEN 3,8,15,D$:GOTO 3160
3140 IF A$<>"C" AND A$<>"c" THEN GOTO 3090
3150 OPEN 3,1,1,D$
3160 PRINT #3,L
3170 FOR Z=0 TO L
3180    PRINT #3,PEEK(D+Z)
3190 NEXT Z
3200 RETURN
3210 REM
3220 REM ******************************
3230 REM * CAMBIAR PUNTERO DE MEMORIA *
3240 REM ******************************
3250 REM
3260 GOSUB 4490
3270 INPUT "MEMORY = ";M
3280 IF M>65535! OR M<0 THEN GOTO 3260
3290 GOSUB 4490
3300 LET MP=M
3310 RETURN
3320 REM
3330 REM ******************
3340 REM * CAMBIO DE BASE *
3350 REM ******************
3360 REM
3370 POKE 214,5:POKE 211,18:PRINT "1. BASE 2."
3380 POKE 214,7:POKE 211,18:PRINT "2. BASE 10."
```

```
3390 POKE 214,9:POKE 211,18:PRINT "3. BASE 16."
3400 GET K$
3410 IF K$="" THEN GOTO 3400
3420 IF K$<>"1" AND K$<>"2" AND K$<>"3" THEN GOTO 3400
3430 LET BA=-(2 AND K$="1")-(10 AND K$="2")-(16 AND K$="3")
3440 LET MO=VAL(K$)
3450 LET K$=""
3460 RETURN
3470 REM
3480 REM ***************
3490 REM * MUEVE BYTES *
3500 REM ***************
3510 REM
3520 GOSUB 4490
3530 INPUT "DESDE? ";P
3540 IF P<0 OR P>65535! THEN GOTO 3520
3550 GOSUB 4490
3560 INPUT "HASTA? ";D
3570 IF D<0 OR D>65535! THEN GOTO 3550
3580 GOSUB 4490
3590 INPUT "LONGITUD? ";L
3600 IF L<0 OR L>65535! THEN GOTO 3580
3610 FOR I=0 TO L-1
3620     POKE D+L,PEEK(P+L)
3630 NEXT I
3640 RETURN
3650 REM
3660 REM *****************************
3670 REM * PASO DE DECIMAL A BINARIO *
3680 REM *****************************
3690 REM
3700 LET P$=""
3710 IF NU>=128 THEN LET P$="1":LET NU=NU-128:GOTO 3730
3720 P$="0"
3730 IF NU>=64 THEN LET P$=P$+"1":LET NU=NU-64:GOTO 3750
3740 P$=P$+"0"
3750 IF NU>=32 THEN LET P$=P$+"1":LET NU=NU-32:GOTO 3770
3760 P$=P$+"0"
3770 IF NU>=16 THEN LET P$=P$+"1":LET NU=NU-16:GOTO 3790
3780 P$=P$+"0"
3790 IF NU>=8 THEN LET P$=P$+"1":LET NU=NU-8:GOTO 3810
3800 P$=P$+"0"
3810 IF NU>=4 THEN LET P$=P$+"1":LET NU=NU-4:GOTO 3830
3820 P$=P$+"0"
3830 IF NU>=2 THEN LET P$=P$+"1":LET NU=NU-2:GOTO 3850
3840 P$=P$+"0"
3850 IF NU=1 THEN LET P$=P$+"1":LET NU=0:GOTO 3870
3860 P$=P$+"0"
3870 RETURN
3880 REM
3890 REM
3900 REM *****************
3910 REM * INSTRUCCIONES *
3920 REM *****************
3930 REM
3940 CLS
3950 PRINT "=============================="
3960 PRINT "** I N S T R U C C I O N E S **"
3970 PRINT "=============================="
3980 PRINT
3990 PRINT "D = DUMP. CAMBIA MODO."
4000 PRINT "B = BUSCA BYTES EN MEMORIA"
4410 NEXT I
4420 LET MP=MP+150
```

```
4430 GOSUB 4490:PRINT "CONTINUAMOS? (S/N)"
4440 GET A$
4450 IF A$="S" OR A$="s" THEN GOTO 4250
4460 IF A$<>"N" AND A$<>"n" THEN GOTO 4440
4470 GOSUB 4490
4480 RETURN
4490 REM
4500 REM ***********************************************
4510 REM * BORRADO DE LOA LINEA 22 Y POSICIONAMIENTO *
4520 REM ***********************************************
4530 REM
4540 POKE 214,22
4550 POKE 211,0
4560 FOR Z=1 TO 40
4570    PRINT " ";
4580 NEXT Z
4590 POKE 214,22
4600 POKE 211,0
4610 RETURN
4010 PRINT "A = AYUDA. PONE ESTAS INSTRUCCIONES."
4020 PRINT "J = JUMP. EJECUTA CODIGO MAQUINA."
4030 PRINT "L = LOAD. CARGA BYTES."
4040 PRINT "M = MEMORY. AJUSTA EL PUNTERO."
4050 PRINT "P = PRINTER. PASA POR IMPRESORA."
4060 PRINT "S = SAVE. GRABA BYTES."
4070 PRINT "T = TRASLADA BYTES."
4080 PRINT "X = CAMBIA DE BASE."
4090 PRINT "2 = AUMENTA EL PUNTERO EN 10."
4100 PRINT "1 = AUMENTA EL PUNTERO EN UNO."
4110 PRINT "9 = DISMINUYE EL PUNTERO EN UNO."
4120 PRINT "8 = DISMINUYE EL PUNTERO EN 10."
4130 PRINT "0 = POKEA EN EL PUNTERO."
4140 PRINT
4150 PRINT "PULSA UNA TECLA"
4160 GET A$
4170 IF A$="" THEN GOTO 4160
4180 PRINT CHR$(147)
4190 RETURN
4200 REM
4210 REM ************************
4220 REM * MODO DUMP DE MEMORIA *
4230 REM ************************
4240 REM
4250 PRINT CHR$(147)
4260 FOR I=MP TO MP+150 STEP 10
4270    IF I>65535! THEN GOTO 4480
4280    LET N$=MID$(STR$(I),2)
4290    IF LEN(N$)<5 THEN LET N$="0"+N$:GOTO 4290
4300    PRINT#CH,N$;": ";
4310    LET Q$=""
4320    FOR J=I TO I+9
4330       LET NU=PEEK(J)
4340       LET M$=CHR$(NU)
4350       IF NU<32 THEN M$="."
4360       GOSUB 2020
4370       Q$=Q$+M$
4380       PRINT#CH,P$;
4390    NEXT J
4400    PRINT#CH,"  ";Q$
```

Los comandos de que disponemos a la hora de utilizar este programa han sido hechos desde IBM pc, por lo que en la pantalla del COMMODORE no aparecerán igual, aunque sí muy parecidas.

Poco hay que explicar sobre el uso del programa, ya que éste es muy sencillo y autoexplicativo. Lo único que hay que decir es que tenemos la posibilidad de ver la pantalla de dos formas distintas. Podemos ver cómo aparece la dirección de memoria junto con el valor que se almacena en dicha posición. Dicho valor podemos verlo en base 10, base 16 e incluso en base 2.

Como muchas veces este formato no es del todo eficiente, hemos incluido otra forma de visualización de la memoria. Vemos cómo aparece la posición de memoria del primer BYTE de cada línea. A continuación vemos diez BYTES en notación hexadecimal y a continuación el carácter que representan en código ASCII. Cuando un carácter no es imprimible, por ser de control, aparecerá como un punto (.). Este modo de visualización de la memoria nos puede servir para buscar mensajes impresos dentro de la misma.

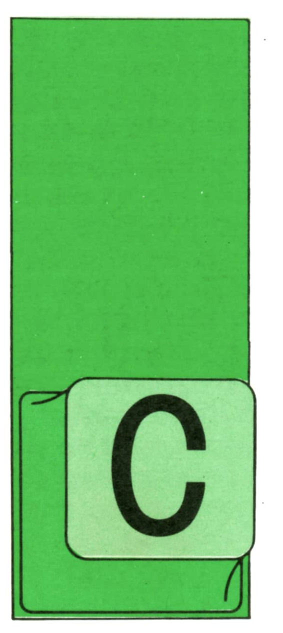
Para cambiar de un modo a otro de visualización, sólo tenemos que pulsar la tecla D.

TECNICAS DE ANALISIS

PROYECTO DE AUTOMATIZACION DE LA PRODUCCION



Fase de diagnóstico



CONCLUIREMOS, ante todo, indicando los últimos objetivos que suelen anotarse en el estudio de la mejora de la producción industrial:

Objetivos humanos

Mejora de las condiciones de trabajo.
Mejora de la seguridad (no accidentes).
Mejora de la formación.

Objetivos de gestión

Mejora del control general de los procesos (costes, valor de los stocks, etc.).
Mejora de los tiempos de reacción en caso de perturbación.
Etc.

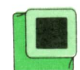


Proyecto de automatización

Entrando ya en el proyecto de automatización de la producción industrial en sí, hemos de anotar que, siguiendo el esquema usual en cualquier proyecto de desarrollo de un producto nuevo a nivel industrial, se suele dividir la tarea en cuatro fases:

— Diagnóstico.
— Análisis y concepción de la solución adecuada.
— Realización.
— Implementación y puesta en funcionamiento.

En estas breves notas, y cara al objetivo que nos ocupa, veremos las características básicas de las dos primeras etapas indicadas (que son las que corresponden a las actividades que se consideran propias de los analistas de los proyectos de automatización industrial).



Diagnóstico

A. *Objetivos del diagnóstico*

Durante la fase de diagnóstico se suelen determinar:

— La estrategia de los medios de personal.
— La estrategia de los medios materiales (células de fabricación, robots, manipuladores, etc.).
— La estrategia del sistema de gestión de la producción (procedimientos, programas informáticos-software, medios de proceso, etc.).
— La estrategia de puesta en funcionamiento de la automatización y de informatización de la producción.

B. *Etapas de la fase de diagnóstico*

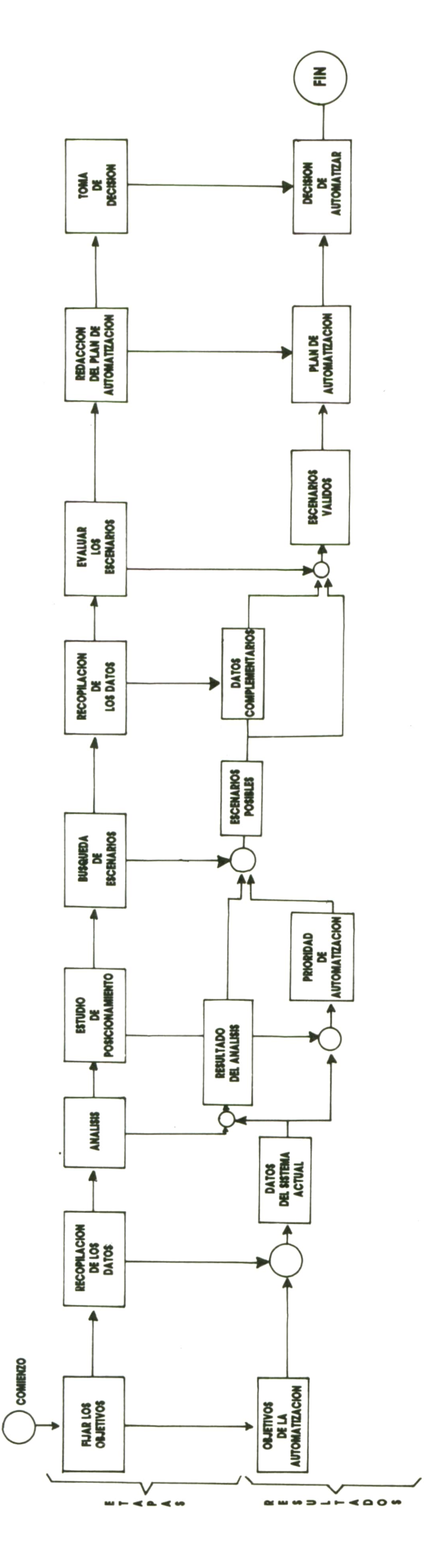
En la figura se presentan, de un modo gráfico, las diferentes etapas que se desarrollan en el proceso de diagnóstico previo al establecimiento del proyecto de mecanización de la producción industrial.

C. *"Herramientas" de ayuda en el diagnóstico*

Se suelen utilizar diferentes "herramientas" para los diversos aspectos involucrados en el diagnóstico:

1. En la *búsqueda de informaciones,* se manejan *indicadores* tanto a nivel de elementos físicos de producción como de control del producto, descriptivos de los centros de producción y sus diferentes unidades, etc. Es importante, en la selección de estos indicadores, asegurarse de que los datos elegidos cumplen las tres características básicas que deben cumplir, y que son: que sean cuantificables, medibles y programables.

2. En el preanálisis propiamente dicho, para la identificación de disfunciones se suelen utilizar las rejillas del método GRAI (que veremos más detalla-



damente al describir las técnicas de análisis); para la localización de "cuellos de botella", un cuadro de relación cargas-capacidades o algún sistema de simulación (aunque en esta fase previa debe considerarse cuidadosamente la relación coste-rendimiento de cualquier técnica aplicada).

3. La fase de *toma de decisiones,* en cuanto al diagnóstico previo, supone la consideración de elementos de inversiones a efectuar (capitales invertidos, ciclo de vida de los productos involucrados, valores residuales, tasas de actualización, etc.), de selección de acciones a tomar, etc. En esta etapa es usual la utilización de algún software específico diseñado para esta tarea; para el empleo de estos programas hay que seguir los siguientes pasos:

— Preparar un fichero que contenga objetos evaluados según un conjunto de criterios preestablecidos.

— Seleccionar los criterios utilizables en el caso presente.

— Definir especificaciones sobre los criterios, cara a eliminar todo objeto que no satisfaga las condiciones presentes.

— Elegir un conjunto por criterio de reflexión.

— Ponderar los criterios en función de su importancia relativa.

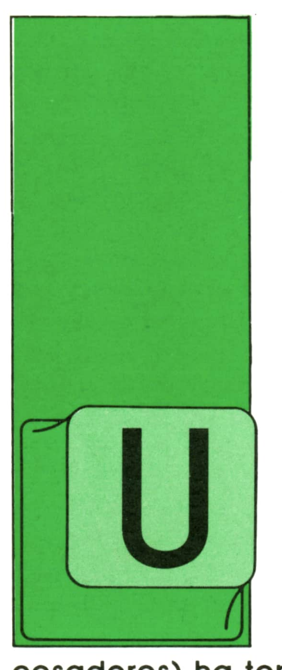
— Modificar la función de evaluación propuesta, observando el efecto de esta modificación sobre los valores tomados por los objetos y su clasificación.

— Llegados a una situación satisfactoria, utilizar la función de evaluación, reteniendo todos los objetos no eliminados y clasificándolos según un valor decreciente.



Fig. 1. Etapas de desarrollo del diagnóstico.

TECNICAS DE PROGRAMACION



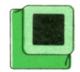


Programación en paralelo

UNA de las técnicas de punta de la Informática moderna es la programación en paralelo. El abaratamiento de los dispositivos de cálculo automático (los microprocesadores) ha tenido como consecuencia que hoy sea factible el diseño de computadoras mucho más complejas que las tradicionales, que disponen de varias unidades centrales de proceso idénticas y conectadas entre sí. La utilización óptima y correcta de estos "sistemas de cálculo en paralelo" o "procesadores vectoriales" dista mucho de estar clara. Las nuevas técnicas han de ir desarrollándose lentamente y con gran esfuerzo. En otras palabras, el "software" parece haberse quedado atrás respecto al "hardware".

¿Por qué ha ocurrido esto? Porque, con muy pocas excepciones, los lenguajes de programación tradicionales no estaban preparados ni son adecuados para enfrentarse a los desafíos impuestos por la nueva forma de programar. En efecto, casi todos ellos son demasiado poco potentes y se basan en la suposición de que el ordenador donde han de actuar los programas ejecuta las instrucciones de una en una, secuencialmente, sin poder realizar dos operaciones a la vez. Sin embargo, esta suposición ya no es aplicable a las nuevas máquinas vectoriales.

Existe una excepción importante a la afirmación anterior: entre los lenguajes de alto nivel hay uno que se presta extraordinariamente a la programación en paralelo. Se trata del lenguaje APL. Vamos a dedicar este capítulo a la presentación de algunos ejemplos que lo demuestran y que servirán de indicio de cómo puede ser la forma de programar del futuro que hará un uso óptimo de las potencialidades de las nuevas máquinas.



Programación lógica en paralelo

La Lógica fue tradicionalmente una parte de la Filosofía, aunque a partir del siglo XIX se convirtió en una de las ramas de las Matemáticas. En su forma más sencilla, la Lógica estudia la verdad o falsedad de las proposiciones (afirmaciones o negaciones) y la forma en que, a partir de algunas de ellas, pueden deducirse otras de las que también podremos afirmar si son verdaderas o falsas.

Veamos un ejemplo de proposición verdadera:

"Todos los hombres son mortales"

Veamos un ejemplo de proposición falsa:

"Todos los hombres son amarillos"

En cambio, la siguiente proposición es verdadera:

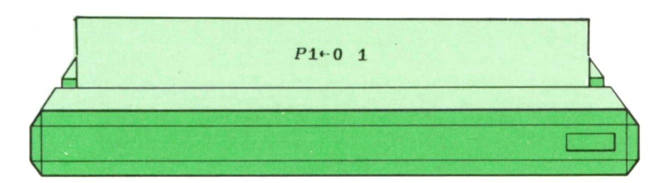
"Algunos hombres son amarillos"

Una de las ramas más populares de la Informática (la "Inteligencia Artificial") se ocupa de la emulación en un ordenador de los procesos mentales humanos. A pesar de que el atractivo del tema consigue con frecuencia titulares de prensa, la Inteligencia Artificial está bastante menos avanzada de lo que podría deducirse de la simple lectura de dichos titulares. Sin embargo, hay ciertas operaciones aparentemente inteligentes (como la solución de problemas lógicos sencillos) que pueden realizarse sin demasiados problemas por medio de un programa de ordenador.

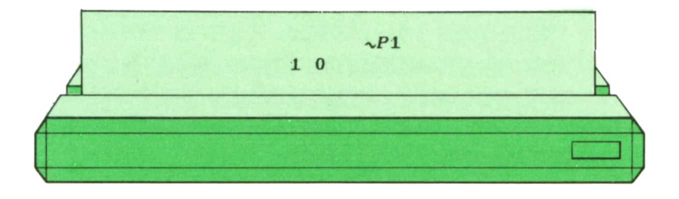
En el lenguaje APL, la verdad o la falsedad de una proposición se representa con los números 0 y 1. El 1 significa que

la proposición es verdadera; el 0, que es falsa.

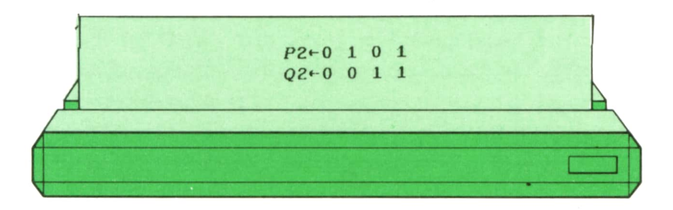
Sea P una proposición cualquiera. Si P es verdadera, diremos que su valor de verdad es 1; si es falsa, diremos que es 0. Sea P1 una variable que representa todos los posibles valores de verdad o falsedad de la proposición P (es decir, 0 y 1). Le asignaremos esos valores de la siguiente manera:



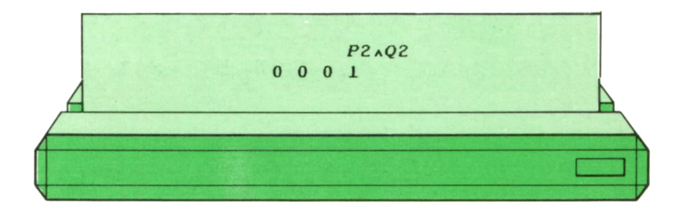
Llamaremos "negación" de una proposición a otra proposición que afirma lo contrario que dicha proposición. Por ejemplo, la negación de "Todos los hombres son mortales" es "No todos los hombres son mortales". Si una proposición es verdadera, su negación será falsa, y viceversa. Veamos cómo se expresa eso en APL:



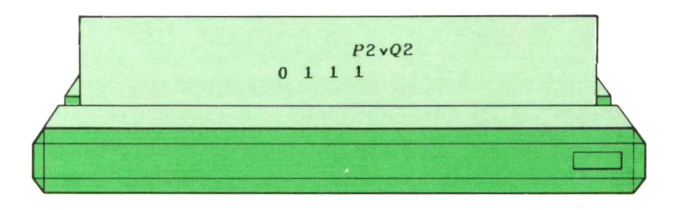
Supongamos ahora que tenemos no una, sino dos proposiciones, P y Q. Vamos a definir dos variables, P2 y Q2, que contengan todas las posibles combinaciones de verdad y falsedad de P y Q. Estas combinaciones son cuatro (P y Q son a la vez verdaderas; a la vez falsas; P es verdadera y Q es falsa; P es falsa y Q verdadera). En consecuencia, los valores que hemos de asignar a P y Q son los siguientes:



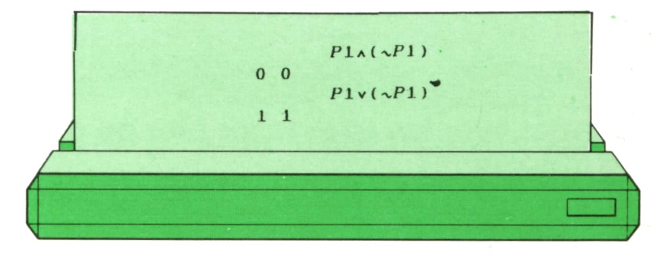
Obsérvese que las columnas del programa anterior nos dan, precisamente, las cuatro combinaciones.



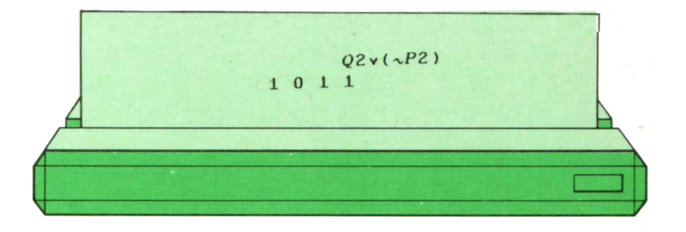
Ahora podemos realizar ciertas operaciones sobre los valores de verdad de P y Q. Por ejemplo, la anterior expresión (que se lee "P y Q") es verdad sólo cuando P y Q son verdad a la vez. Mientras que la siguiente expresión (que se lee "P o Q" sólo es falsa cuando P y Q son falsas a la vez:



Por consiguiente, una proposición y su negación nunca serán verdad a la vez, mientras que siempre se cumplirá una o la otra. Dicho de otro modo: "P y no P" no pueden ser verdad a la vez, mientras que "P o no P" siempre es cierto:



Decimos que una proposición P implica otra, Q, cuando Q es verdad o P es falso.



Obsérvese que la expresión anterior tiene un solo 0) que corresponde al caso en que P es verdad y Q es falso. Es decir: "P implica Q es falso sólo si P es verdad y Q es falso", como era de esperar.



Solución de un problema lógico

Se habrá observado que, en todos los ejemplos anteriores, una sola expresión APL permite obtener de un solo golpe (en paralelo) todas las soluciones posibles que corresponden a las distintas combinaciones de valores de las variables lógicas con las que estamos trabajando. Ahora vamos a ver cómo se puede emplear esta técnica para resolver un problema lógico algo más complicado.

El problema en cuestión está sacado del libro "¿Cómo se llama este libro?", de Raymond Smullyan (Ediciones Cátedra, 1978). El problema se enuncia así:

"Cuando Alicia entró en el bosque del olvido no lo olvidó todo, solamente ciertas cosas. A menudo olvidaba su nombre, y una de las cosas que más disposición tenía a olvidar era el día de la semana. Ahora, el León y el Unicornio visitaban frecuentemente el bosque. Los dos son criaturas extrañas. El León miente los lunes, martes y miércoles y dice la verdad los otros días de la semana. El Unicornio, por otra parte, miente los jueves, viernes y sábados, pero dice la verdad los restantes días de la semana."

"Un día Alicia se encontró con el León y el Unicornio que descansaban bajo un árbol. Ellos dijeron lo siguiente":

"León: Ayer fue uno de los días en que me toca mentir."

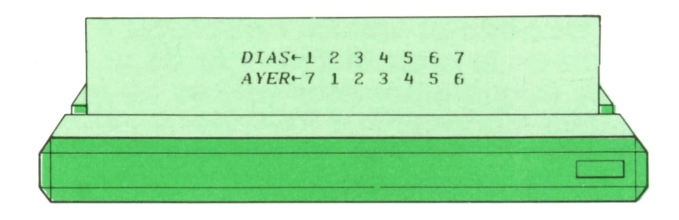
"Unicornio: Ayer fue uno de los días en que me toca mentir."

"A partir de estas dos frases, Alicia (que era una chica lista) fue capaz de deducir el día de la semana. ¿Qué día era éste?"

Vamos a resolver el problema mediante programación lógica en paralelo. En primer lugar, es preciso definir los datos. La variable DIAS contendrá los siete días de la semana, representados por los números del 1 al 7, de acuerdo con la siguiente tabla:

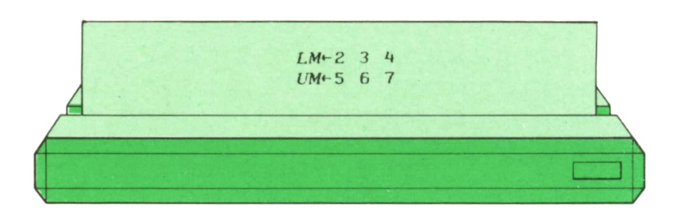
1: Domingo.
2: Lunes.
3: Martes.
4: Miércoles.
5: Jueves
6: Viernes
7: Sábado

Por otro lado, la variable AYER contendrá el día anterior a cada uno de los días indicados (donde 7 es el día anterior a 1). Estas dos variables se definirán, en APL, de la siguiente manera:



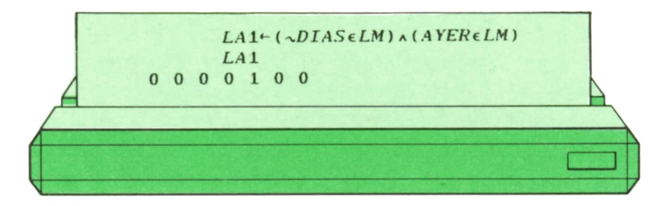
Obsérvese que cada elemento de ayer es, efectivamente, el día anterior al elemento correspondiente de DIAS.

Necesitamos también dos variables que nos digan qué días miente el León (LM) y qué días miente el Unicornio (UM):



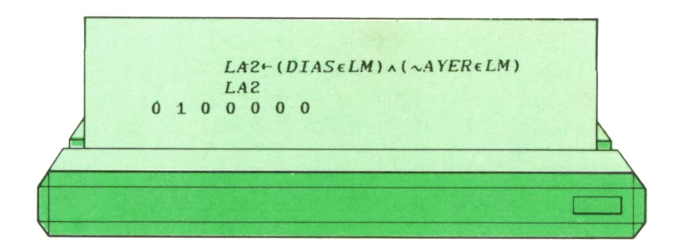
Ya tenemos los datos. Ahora vamos a resolver el problema. Para ello tendremos que escribir expresiones lógicas que estén de acuerdo con el enunciado del problema. Veamos cómo:

El León ha dicho: "Ayer fue uno de los días en que me tocaba mentir." Esto lo puede decir en dos casos distintos: si hoy dice la verdad y ayer mentía, o si hoy miente y ayer dijo la verdad. El primer caso puede expresarse así:



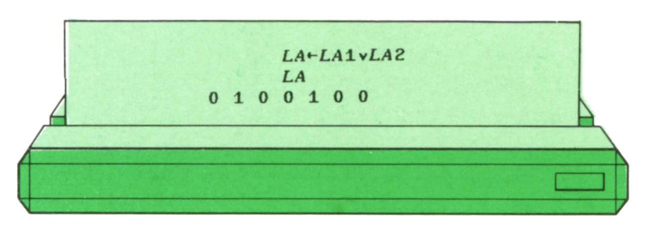
que puede leerse de esta manera: "Días que no pertenecen a los que el León miente, mientras que Ayer pertenece a los que el León miente." Obsérvese que el resultado nos indica que esto sólo puede ocurrir en jueves. Hemos guardado este resultado en la variable LA1 (Alternativa 1 del León).

El segundo caso (el León miente hoy, pero ayer dijo la verdad) puede expresarse así:

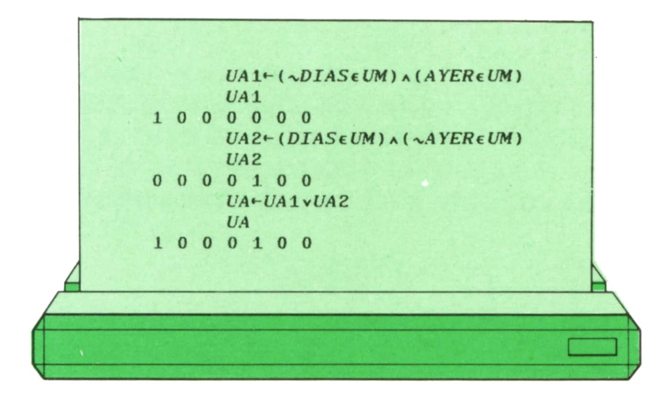


que puede leerse de esta manera: "Días que pertenecen a los que el León miente, mientras que Ayer no pertenece a los que el León miente." En este caso, el resultado nos indica que esto sólo puede ocurrir en lunes. Hemos guardado este resultado en la variable LA2 (Alternativa 2 del León).

Cualquiera de las dos posibilidades es aceptable. Por tanto, debemos unir las dos expresiones mediante la operación "O". Guardemos el resultado conjunto de la variable LA (alternativas del León):

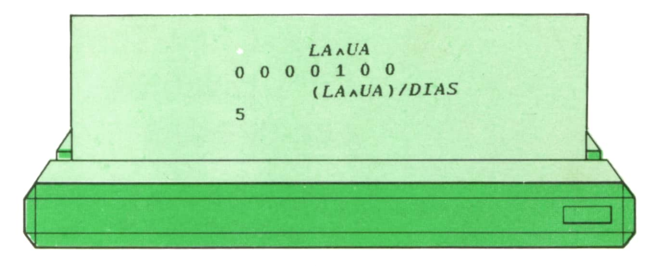


Ahora hacemos lo mismo para el Unicornio (que dijo la misma frase):



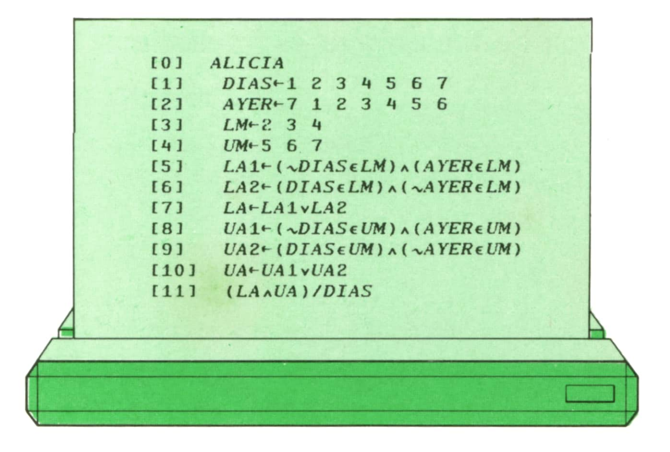
Obsérvese que la primera alternativa del Unicornio (variable UA1) sólo puede ocurrir en domingo, mientras que la segunda (variable UA2) sólo puede ocurrir en jueves.

Ahora bien: las frases pronunciadas por el León y el Unicornio deben ser posibles a la vez (eso es lo que dice el enunciado del problema). Por tanto, debemos unir con la operación "Y" las dos variables LA y UA, que nos dan las situaciones en que son posibles las frases del León y del Unicornio.

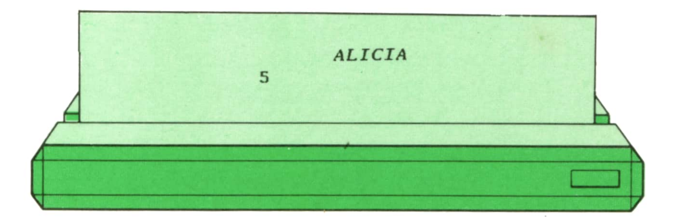


Puede verse con facilidad que la conjunción de ambas frases sólo es posible en el día 5 (el jueves).

Resumamos todo lo anterior en un solo programa, que resuelve el problema lógico propuesto:

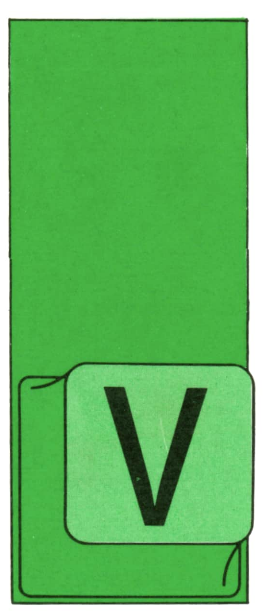


Veamos ahora su ejecución completa:



Se observará que el método utilizado consiste en probar simultáneamente todas las soluciones posibles (los siete días de la semana) y quedarse, finalmente, con la que cumple las condiciones especificadas por el enunciado del problema. Este método es especialmente útil cuando se desea aprovechar las facilidades de la programación en paralelo.

LOGO



Cómo formar palabras compuestas

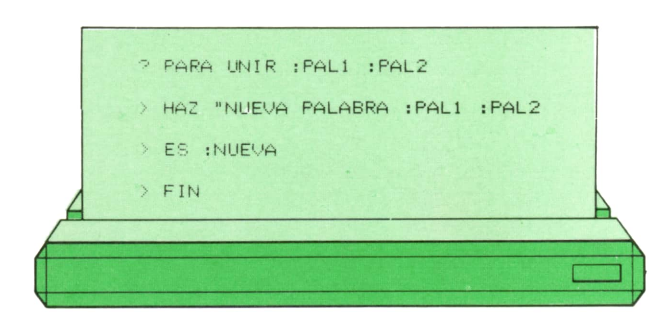
VAMOS a escribir un procedimiento muy sencillo que nos va a permitir formar palabras compuestas.

Para eso necesitamos utilizar la función

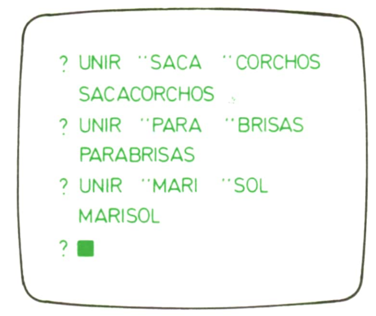
PALABRA p1 p2

que devuelve la palabra que se obtiene al unir las letras o palabras p1 y p2.

El procedimiento es el siguiente:



Si lo probamos nos queda:





Cómo leer una lista

Ya sabemos que al utilizar la función LC la tortuga se queda esperando a que pulsemos una tecla y cuando lo hacemos, guardamos su valor (una letra o un número) en una variable. Pero con ella no tenemos la posibilidad de leer palabras o listas.

Para ello, hay que usar la función

LEELISTA

o en abreviatura

LL

Esta función hace que la tortuga nos deje escribir en la pantalla una palabra o una frase y se quede sin hacer nada hasta que pulsamos la tecla RETURN. Cuando lo hacemos, convierte lo que hayamos escrito en una lista.

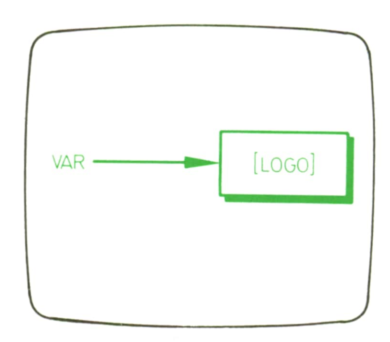
Como siempre, esa lista la almacenaremos en una variable poniendo

HAZ "VAR LL

Hay que tener cuidado a la hora de utilizar LL para leer una palabra. Si escribimos



en la variable VAR se nos almacena



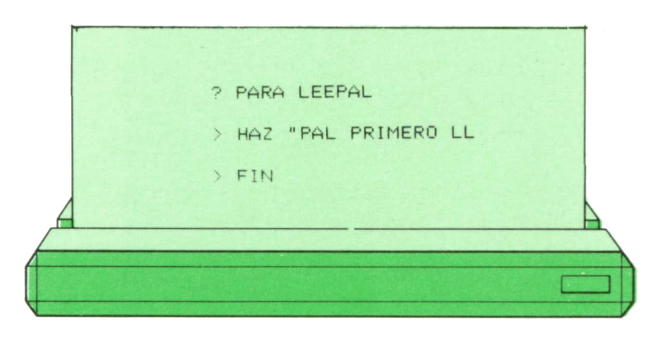
es decir, una lista formada por una sola letra. Si queremos que la tortuga guarde tenemos que utilizar una función nueva. Esta función es

PRIMERO 1

```
?HAZ "COLORES [ROJO AMARILLO VERDE]
?ES PRIMERO :COLORES
 ROJO
?ES ULTIMO :COLORES
 VERDE
?ES MP :COLORES
 AMARILLO VERDE
?ES MU :COLORES
 ROJO AMARILLO
?■
```

que devuelve el primer elemento de la lista 1.

Por tanto, podemos escribir un procedimiento que nos permita leer palabras y guardarlas en una variable. Es el siguiente:



Otras funciones para manejar listas

Vamos a ver algunas funciones que se usan para trabajar con listas. Como siempre, el resultado que nos devuelva la tortuga lo tendremos que utilizar con algún comando.

La función

ULTIMO 1

devuelve el último elemento de la lista *1*, mientras que

MENOSPRIMERO 1

o en abreviatura

MP 1

da como resultado una lista quitando el primer elemento de la lista *1* y

MENOSULTIMO 1

o en abreviatura

MU 1

da una lista quitando el último elemento de *1*.

```
?ES FRASE "BUENAS "TARDES
 BUENAS TARDES
?ES PP "AZUL :COLORES
 AZUL ROJO AMARILLO VERDE
?ES PU "AZUL :COLORES
 ROJO AMARILLO VERDE AZUL
?■
```

Por otro lado, la función

FRASE p1 p2

devuelve una lista formada por las palabras p1 y p2, mientras que las funciones

PONPRIMERO p 1

o en abreviatura

PP p 1

y

PONULTIMO p 1

o en abreviatura

PU p 1

devuelven una lista que se obtiene al añadir la palabra *p* a la lista *1* al principio o al final, respectivamente.

```
?ES CUENTA :COLORES
 3
?ES ELEMENTO 2 :COLORES
 AMARILLO
?ES VACIO? :COLORES
 FALSO
?ES VACIO? [ ]
 CIERTO
?■
```

Por último, existen tres funciones un poco especiales. La función

CUENTA 1

devuelve el número de elementos que tiene la lista *1*.

Con la función

ELEMENTO n 1

la tortuga nos da el elemento que ocupa la posición *n* dentro de la lista *1* y con

VACIO? 1

nos dice si la lista *1* contiene (devuelve CIERTO) o no (devuelve FALSO) algún elemento.



Cómo tener una agenda de teléfonos

Vamos a escribir varios procedimientos con los que podremos hacer varias cosas con una agenda que nos servirá para guardar los teléfonos de nuestros amigos. Podremos crearla, añadir un nombre con su teléfono, buscar el teléfono correspondiente a un nombre en concreto o ver la lista de nombres y teléfonos que tengamos en la agenda.

El procedimiento general se llama AGENDA y se encarga de utilizar el resto. Tenemos que escribir:

```
? PARA AGENDA
> TEXTO
> BT
> ES [ELIGE UNA OPCION:]
> ES [] ES [] ES []
> ES [1 EMPEZAR] ES []
> ES [2 AÑADIR] ES []
> ES [3 CONSULTAR] ES []
> ES [4 VER TODO] ES []
> ES [0 TERMINAR] ES []
> ES [] ES []
> TECLEA "OPCION:
> HAZ "TECLA LC
> ES :TECLA
>  SI  NO MIEMBRO?  :TECLA [0 1 2 3 4] [ES [TECLA NO  VALIDA]  ES
[PULSA CUALQUIER TECLA PARA CONTINUAR] HAZ "TECLA LC AGENDA]
> SI :TECLA = 0 [ALTO]
> SI :TECLA = 1 [EMPEZAR]
> SI :TECLA = 2 [AÑADIR]
> SI :TECLA = 3 [CONSULTAR]
> SI :TECLA = 4 [VER]
> ES [] ES []
> ES [PULSA CUALQUIER TECLA PARA CONTINUAR]
> HAZ "TECLA LC
> AGENDA
> FIN
? PARA EMPEZAR
> HAZ "NOMBRES []
> HAZ "TELEFONOS []
> AÑADIR
> FIN
? PARA AÑADIR
> BT
> TECLEA "NOMBRE:
> HAZ "NOMB PRIMERO LL
> ES []
```

```
> TECLEA "TELEFONO:
> HAZ "TELEF PRIMERO LL
> HAZ "NOMBRES PONULTIMO :NOMB :NOMBRES
> HAZ "TELEFONOS PONULTIMO :TELEF :TELEFONOS
> FIN
? PARA CONSULTAR
> BT
> TECLEA "NOMBRE:
> HAZ "NOMB PRIMERO LL
> ES []
> HAZ "NUM 1
> BUSCAR
>  SI  :NUM  >  CUENTA :NOMBRES [ES [ESTE NOMBRE NO  ESTA  EN  LA
AGENDA] ALTO]
> TECLEA [EL TELEFONO ES:]
> ES ELEMENTO :NUM :TELEFONOS
> FIN
? PARA BUSCAR
> SI :NUM > CUENTA :NOMBRES [ALTO]
> SI ELEMENTO :NUM :NOMBRES = :NOMB [ALTO]
> HAZ "NUM :NUM + 1
> BUSCAR
> FIN
? PARA VER
> BT
> TECLEA [LA AGENDA TIENE -]
> TECLEA CUENTA :NOMBRES
> ES [- ELEMENTOS]
> ES [] ES [] ES []
> RECORRER :NOMBRES :TELEFONOS
> FIN
? PARA RECORRER :N :T
> SI VACIO? :N [ALTO]
> TECLEA PRIMERO :N
> TECLEA [< - - - >]
> ES PRIMERO :T
> RECORRER MP :N MP :T
> FIN
```

en donde NOMBRES Y TELEFONOS son dos variables en las que vamos almacenando la lista de los nombres de nuestros amigos y la de sus teléfonos.

NOTA: el comando

BT

nos permite borrar el texto que haya en la pantalla.



Os proponemos

1. Añade más opciones al procedimiento AGENDA y los procedimientos necesarios para poder hacer las siguientes cosas:

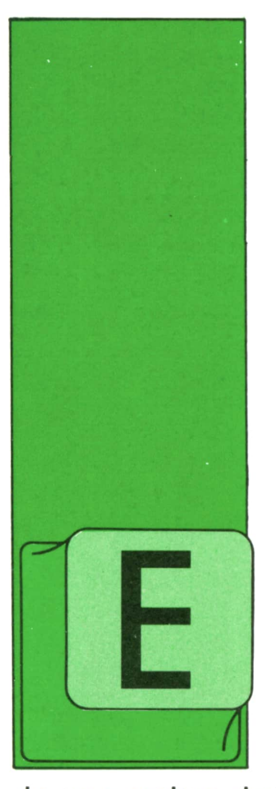
a) Borrar un nombre y su teléfono de la agenda.

b) Cambiar el número de teléfono correspondiente a un nombre.

c) Comprobar que cuando se añade un nombre en la agenda, éste no está ya en la misma (para que un nombre no esté dos veces).

d) Clasificar los nombres y sus teléfonos por orden alfabético.

PASCAL





La búsqueda dicotómica

En multitud de ocasiones se plantea el problema de encontrar, en una lista ordenada de elementos, uno de ellos en concreto; pensemos, por ejemplo, en la búsqueda de un nombre dado en una página de la guía telefónica.

En casos así, una posible forma de proceder es empezar observando el primero e ir avanzando de uno en uno hasta encontrar el deseado. Si la lista tuviese N elementos, el número de observaciones necesario para encontrar un elemento cualquiera sería, en valor medio, N dividido por 2.

Sin embargo, y suponiendo que la única característica conocida de la lista sea su criterio de ordenación, el mejor sistema de búsqueda, es decir, el que en término medio permite encontrar el elemento con un menor número de observaciones, es el denominado método de búsqueda dicotómica, que consiste en lo siguiente:

El primer elemento que se observa es el del medio de la lista; si coincide con el buscado, el problema queda resuelto, pero si no, queda reducido a encontrar el elemento en una de las dos medias listas que delimita el elemento central; según como sean las posiciones relativas del elemento central y del elemento buscado, resulta posible determinar en cuál de las dos mitades hay que seguir buscando. Con la media lista seleccionada el proceso se repite: se observa el elemento central y, o bien se acaba el problema, o bien queda reducido a buscar en una lista con la cuarta parte de elementos, etc. La forma habitual de buscar algo en una guía telefónica por tanteos es más o menos parecida a esto.

Supongamos que la lista tenga 256 elementos y que el buscado sea el de la posición 183. En primer lugar, observaríamos el 128 y, al estar entre el 128 y el 256; a continuación observaríamos en el medio de ese tramo, es decir, en la posición 192, y decidiríamos que ha de estar entre 128 y 192, por lo que pasaríamos a observar el 160, etc. La secuencia sería:

```
128   192   160   176   184   180
182   183
```

Habría que hacer, por tanto, ocho observaciones en el peor de los casos y aproximadamente siete en valor medio; en concreto, para una lista con N elementos el máximo número de observaciones necesario sería el logaritmo en base 2 de N o, dicho de otra forma, con X observaciones sería posible encontrar cualquier elemento de una lista con 2 elevado a X elementos.

Como ejemplo concreto veamos cómo obtener en un tabla de números situados en orden creciente la posición de uno dado. Supongamos que la tabla se define así:

```
const
  Total = 1000;
var
  Tabla: array (1..Total) of integer;
```

Una función que devolviese la posición de un número obtenida por búsqueda dicotómica, o un cero en caso de no encontrarlo, podría ser:

```
function Posicion (Num: integer): integer;
{----------------------------------------------------}
{ Busca en Tabla la posicición de Num y la devuelve. }
{ Retorna un cero si no lo encuentra.                 }
{----------------------------------------------------}
 var
   Primero,              { delimita el tramo donde buscar }
   Ultimo,               { delimita el tramo donde buscar }
   EnMedio : integer;    { indica el elemento a observar  }
   Ok      : boolean;

begin
 Primero:= 1;
 Ultimo := Total;                     { se empieza con toda la lista }
 Ok     := false;

{ mientras haya donde buscar: }

 while (Primero <= Ultimo) and not Ok do
   begin
    EnMedio:= (Primero + Ultimo) div 2;
    if Tabla [EnMedio] < Num then Primero:= EnMedio + 1
     else
       if Tabla [EnMedio] > Num then Ultimo := EnMedio - 1
        else
          Ok:= true
   end;

 if Ok then Posicion:= EnMedio else Posicion:= 0
end;
```

Búsqueda dicotómica de raíces de polinomios con el método de Sturm

Recordemos el método de Sturm de obtención de raíces reales de polinomios; con él era posible construir una función entera, a la que llamábamos CAMBIOS DE SIGNO, que servía para determinar la cantidad de raíces reales existentes entre dos puntos cualesquiera. En concreto, las raíces se encontraban justo donde la función presentaba sus escalones o cambios de valor (aunque no se ha mencionado, como el lector supondrá, la presencia de un escalón de dos unidades, por ejemplo, corresponde a una raíz doble, puesto que la diferencia en número de cambios de signo entre un punto situado inmediatamente antes del escalón y otro situado inmediatamente después es dos).

Como los escalones van siempre en el mismo sentido (siendo máximo el número de cambios de signo en menos infinito), a la hora de buscar la ubicación exacta de un escalón es posible utilizar una exploración dicotómica.

Supongamos que sabemos que en 0 los cambios de signo son 3, que en 10 son 2 y que, por tanto hay una raíz simple entre medias. En primer lugar, observaríamos los cambios de signo que se producen en el punto intermedio 5; si éstos fuesen 3, llegaríamos a la conclusión de que el escalón se encuentra entre 5 y 10, por lo que el siguiente punto a explorar sería 7, 5, etc. Con diez evaluaciones sucesivas de los cambios de signo, el intervalo inicial de 10 unidades se vería reducido a uno de 0,01, aproximadamente.

Al ser números reales, el proceso de estrechamiento del cerco en torno a una raíz en teoría podría seguir indefinidamente; por tanto, habría que poner como condición para acabar la búsqueda el que el tamaño del intervalo fuese menor que una cierta cantidad, cantidad que no ha de ser menor que los posibles errores de redondeo con variables reales para evitar que, por esta causa, nunca se cumpla la condición.

Veamos qué procedimientos añadir al programa de Sturm para permitir la búsqueda automática de raíces:

```
{--------------------------------------------------------------------}
procedure BUSCA_RAICES (var S: Secuencia_t);
{-----------------------------------------------------------------------}
{  Localiza las raices reales en un intervalo dado, de menor a mayor   }
{-----------------------------------------------------------------------}
  const
   Error = 3e-10; { máximo intervalo admisible, en tanto por uno de X }

  var
   Izquierda,
   Derecha,
   Pie_De_Escalon,
   Raiz                   : real;

   Cambios_Izquierda,
   Cambios_Derecha,
   Salto                  : integer;

   {-------------------------------------------------------------------}
   function X_CON_ESCALON: real;
   {-------------------------------------------------------------------}
   {  Busca X de primer escalon empezando por Izquierda, por dicotomía }
   {  Cierra el cerco entre Izquierda y Pie_de_Escalon.                }
   {-------------------------------------------------------------------}
      var X:real; Cont: byte;
    begin
     Pie_De_Escalon := Derecha;
     Cont := 0;
     repeat
      X     := (Izquierda + Pie_De_Escalon) / 2.0;
      if Cambios_de_Signo (S,X) < Cambios_Izquierda then Pie_De_Escalon:= X
                                                    else Izquierda      := X;
      Cont := Cont + 1;
     until ((Pie_De_Escalon - Izquierda) < Error * abs (X))
                                { esto puede dar problemas cuando una raiz }
                                { sea cero, por eso está el contador:      }
           or (Cont = 100);     { 100 es suficiente, pues  2^100 = 1.26e30 }
     X_Con_Escalon := X
    end;
   {-------------------------------------------------------------------}

 begin
  writeln ('DEFINICION DE INTERVALO DE BUSQUEDA:'): writeln;
  write ('Límite inferior = '); readln (Izquierda);
  write ('Límite superior = '); readln (Derecha);
  writeln;

  Cambios_Izquierda:= Cambios_de_Signo (S,Izquierda);
  Cambios_Derecha   := Cambios_de_Signo (S,Derecha);

  writeln ('Desnivel entre -infinito e izquierda = ',
           Cambios_en_Infinito (S,-1) - Cambios_Izquierda);
  writeln ('Desnivel en intervalo                = ',
           Cambios_Izquierda - Cambios_Derecha);
  writeln ('Desnivel entre derecha y +infinito   = ',
           Cambios_Derecha - Cambios_en_Infinito (S,+1));
  writeln;
 {----------------------------------------------}
 while Cambios_Izquierda > Cambios_Derecha do
   begin
    Raiz := X_con_Escalon;
    Salto:= Cambios_Izquierda - Cambios_de_Signo (S, Pie_De_Escalon);

    writeln ('X      = '  ,Raiz);
    writeln ('Salto =    ',Salto);
    writeln ('P(X)   = '  ,Valor (S[0],Raiz));
    writeln;
                                    { recorta intervalo por izquierda: }
    Izquierda          := Pie_De_Escalon;
    Cambios_Izquierda := Cambios_Izquierda - Salto
   end;

 write ('Pulse Intro para seguir.'); readln
end;
```

La parte principal del programa debe ser modificada para ampliar las opciones del menú:

```
begin
 repeat
  writeln;
  writeln ('1 - Introducir polinomio.');
  writeln ('2 - Mostrar Secuencia de Sturm.');
  writeln ('3 - Búsqueda de raíces.');
  writeln ('4 - Búsqueda automática.');
  writeln ('5 - Salida del programa.');
  writeln;
  write ('ESCOJA OPCION:  '); readln (Ca);
  writeln;

  case Ca of

   '1': begin                          { entrada de datos }
         Lee_Pol (Pol, 'P');
         Obtener_Secuencia (Pol, Sec)
        end;

   '2': for I := 0 to Pol.Grado do     { mostrar secuencia de Sturm }
          begin
           writeln ('------------S[',I,']-------------');
           Saca_Pol (Sec[i], 'S');
           writeln;
           write ('Pulse Intro para seguir'); readln
          end;

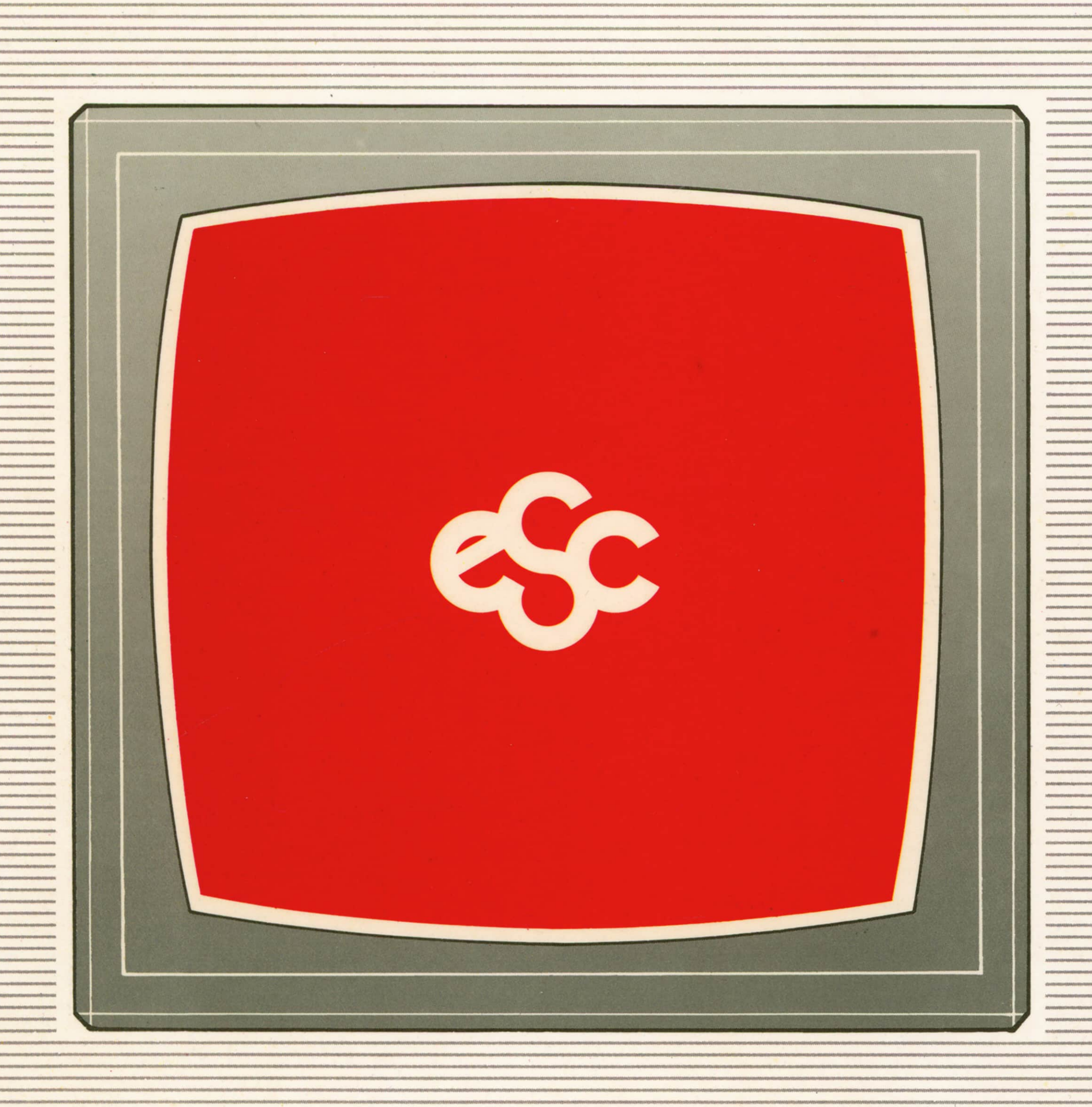
   '3': begin                        { Búsqueda de raíces }
         writeln ('Número de cambios de signo en - infinito = ',
                   Cambios_en_Infinito (Sec, -1));
         writeln ('Número de cambios de signo en + infinito = ',
                   Cambios_en_Infinito (Sec, +1));
         repeat
           writeln;
           write ('X    = '); read (X);
           for I:=0 to Pol.Grado do
             begin
              writeln;
              write ('S[',i,']= ',Valor (Sec[i], X))
             end;
           writeln ('           NUMERO DE CAMBIOS DE SIGNO = ',
                     Cambios_de_Signo (Sec, X));
           writeln;
           write ('¿Desea seguir? (S/N) '); readln (Ca);
         until (Ca = 'N') or (Ca = 'n')
        end;

   '4': Busca_Raices(Sec)

   end;

 until Ca = '5'

end.
```



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼